파네시아 기술리포트

AI 인프라 혁신의 중심,
메모리·링크 중심의 연결 반도체와 데이터센터 연결 솔루션

정명수 (주)파네시아 대표이사 | 한국과학기술원 석좌교수

2025-07-22

panmnesia

목차

panmnesia

들어가기 앞서서

- 본 기술 보고서는 “Compute Can’t Handle the Truth: Why Communication Tax Prioritizes Memory and Interconnects in Modern AI Infrastructure[1]”의 한글 직역과 일부 변환 툴을 사용하여 작성된 문서로, 표현상의 검수는 완료하였으나 정확한 문맥과 내용의 이해를 위해서는 원본 영문 기술 보고서를 참조하기 바랍니다.
- 본 기술 보고서는 파네시아 데모 내용과 XLink에 대한 부분을 제외하고 모두 2024년 8월 반도체공학회 하계학술대회의 파네시아 키노트 연설의 내용을 기반으로 작성되었습니다.
- 본 기술 보고서는 학술 연구 결과나 출판물을 위한 것이 아니라, 다양한 독자들의 이해를 돕기 위한 목적으로 작성된 안내 자료입니다. 따라서 학술적 활용보다는 대규모 언어 모델 그리고 AI 인프라, 연결 반도체에 대해 쉽게 이해하고 접근하고자 하는 독자들에게 권장합니다.
- 본 기술 보고서에서 소개된 다수의 내용은 특허로 보호받고 있습니다.

[1]Link: https://panmnesia.com/technology/pub/compute-cant-handle-the-truth-why-communication-tax-prioritizes-memory-and-interconnects-in-modern-ai-infrastructure/

1. 기술 보고서 개요

인공지능(AI, Artificial Intelligence), 특히 기계학습(ML, Machine Learning)은 지난 수십 년간 점진적인 기술 발전과 다양한 알고리즘적 전환을 반복하며 지속적인 성장을 거듭해 왔다 [1–4]. 초기 AI모델과 서비스는 주로 연산 성능 향상에 기반하여 발전하였으나, 최근의 주요 진전은 대규모 데이터의 확보 가능성과 메모리 관리 기술, 인터커넥트 아키텍처의 진보에 크게 의존하고 있다. 이러한 기술들은 오늘날의 AI 시스템이 인간 수준 또는 그 이상의 인지 능력을 모사할 수 있도록 발전하는 것에 큰 기여를 하 있으며, 이로 인해 현대 AI 기술은 이미지 인식, 자연어 이해, 대화형 상호작용, 창의적 콘텐츠 생성 등의 다양한 작업에서 탁월한 성능을 발휘하고 있다 [5–9].

대부분의 AI 기법은 기본적으로 비정형 데이터를 벡터와 행렬 등 구조화된 수치 표현으로 변환하여 고차원 공간에서 복잡한 패턴을 학습하는 방식을 사용한다 [10–14]. 이 과정에서 AI 모델은 고차원 손실함수(Loss Function)의 특성을 효과적으로 활용하기 위해 경사 하강법(Gradient Descent)과 같은 알고리즘적 최적화 기법을 활용하여 수십억에서 수조 개의 파라미터들을 반복적으로 조정하며 학습하는 것이 일반적이다 [15–19]. 이러한 AI 모델의 표현력은 학습에 사용되는 고차원 데이터를 얼마나 정밀하게 내재화할 수 있는가에 달려 있으며, 이에 따라 학습 정확도와 일반화 성능이 결정된다고 볼 수 있다. 따라서, 다양한 영역에서 더 높은 정확도와 성능을 위해 사용하는 데이터의 복잡성과 그 크기가 증가하였으며, 해당 메모리와 연산 자원의 요구량 또한 기하급수적으로 증가하게 되어 기존 중앙 처리 장치(CPU, Central Processing Unit) 중심의 인프라를 벗어나게 되었다.

오랜 기간 잠재되어 있던 AI 연구의 성과가 최근 챗GPT와 같은 실생활 응용으로 급격히 전환된 것은 아날로그에서 디지털로, 필름에서 디지털 카메라로, 내연기관에서 전기차로의 전환과 비견되는 새로운 기술 패러다임의 도래를 의미한다.

이러한 모델 변화에 따라 높아진 연산 능력을 수용하기 위해, 최신 데이터센터(Data Centers)와 고성능컴퓨팅(HPC, High-Performance Computing) 환경에서는 CPU 대신 AI 반도체라고 불리는 하드웨어 가속기(Accelerator)를 포함하여 그래픽 프로세싱 유닛(GPU, Graphics Processing Unit)들이 중요 하드웨어 자원으로 자리 잡았다[2] [20–35]. 특히, GPU는 수천 개의 병렬 연산 유닛과 고대역폭 메모리(HBM, High-Bandwidth Memory)를 통합하여 대규모 AI 연산에 적합하도록 설계되고 진화해 왔다 [22, 36–39]. 다만, 이러한 진화에도 불구하고 GPU 또는 가속기가 보유할 수 있는 메모리의 용량으로는 급격한 AI 기법의 변화와 대규모 모델들을 수용할 수 없는 상태에 도달했다. 예를 들어 라마 3(Llama 3) 405B 모델 [7, 8]과 같이 십만 개 이상의 토큰을 입력으로 받는 경우, 임베딩, 활성화 값, 옵티마이저 상태 등을 저장하기 위해 100테라바이트(TB, Terabyte)가 넘는 대용량의 메모리가 필요하다 [24, 25, 28, 40–46]. 이는 엔비디아(NVIDIA)의 블랙웰(Blackwell)과 같은 최신 GPU [20, 21]가 제공하는 100에서 200 기가바이트(GB, Gigabyte) 수준의 온보드(On-Board) 메모리를 훨씬 상회하는 수치다. 따라서 최근 AI 시스템에서는 수천수만 대의 GPU를 함께 배치하여 모델이나 데이터를 나누어 병렬 실행하는 구조의 사용이 필수가 되었고, 이러한 분산 처리 및 동기화 요구에 따라 발생하는 GPU 간 통신 오버헤드가 전체 학습 시간의 35%–70%에 달할 정도로 전체 AI 인프라의 병목에 중심이 되었다 [32, 47–51]. 다양한 학문 분야에서 연구가 수행되었으나, 수십 년간 우리 생활에 직접 활용되지 못하던 AI 연구의 결과가 오픈AI(OpenAI)의 챗GPT(ChatGPT)와 같은 실제 응용 분야로 전환이 이루어지고, 많은 사람들에게 직접 영향을 주기 시작한 것이 불과 최근 2년 안에 일어난 사실이라는 것을 생각해 보면, 대규모 AI 워크로드에서 발생하는 빈번한 GPU 간 통신, 이로 인한 방대한 데이터 교환, 그리고 높은 메모리 요구사항이 얼마나 현대 AI 인프라 시스템에 부담이 될지 쉽게 유추해 볼 수 있다.

대규모 AI 워크로드에서 발생하는 빈번한 GPU 간 통신, 이에 따른 방대한 데이터 교환, 그리고 높은 메모리 요구사항 등은 AI 인프라와 데이터센터의 주요 과제가 더 이상 단순한 연산량 자체에 국한되지 않음을 보여준다.

다시 말해, 이와 같은 현실은 더 이상 연산량 자체가 AI 인프라에서 중요 지표가 될 수 없으며, 특정한 응용 처리 가속 및 하드웨어 가속기 구축만이 우리가 준비하고 풀어야 하는 과제의 핵심이 될 수 없음을 알려준다. 앞에서 언급되었듯이 대규모 분산, 병렬 처리에서 오히려 핵심 문제는 GPU나 가속기 간의 방대한 데이터 이동, 메모리 사용, 통신 동기화와 관련된 시스템 레벨 성능 저하와 전력 소모라고 볼 수 있다. 기존의 GPU 중심 구조는 메모리 컨트롤러가 프로세서에 밀착 통합되어 있어 외부 확장이 어려우며, PCIe나 NVMe 기반 스토리지를 통한 메모리 접근 시 수백 나노초(ns) 에서 수십 마이크로초 (μs) 수준의 지연이 불가피하게 발생하여 GPU 활용도가 현저히 저하된다 [26, 27, 52, 53]. 이러한 문제를 해결하기 위해, "컴퓨트 익스프레스 링크(CXL, Compute Express Link [54–56])" 기술을 기반으로 하는 연결 반도체들이 차세대 AI 인프라의 핵심으로 부상하고 있다. CXL 인터커넥트(Interconnect) 기술은 메모리 컨트롤러를 연산 유닛으로부터 완벽히 분리하고, 여러 컴퓨팅 노드가 공유할 수

[2] 본 기술 보고서에서 하드웨어 가속기는 일반적인 CPU를 제외한 신경망 프로세서 유닛(NPU, Neural Processing Unit), AI 프로세싱 유닛(APU, AI Processing Unit), 도메인 특화 가속기(DSA, Domain Specific Accelerator) 그리고 GPU 등을 모두 포함한다.

있는 일관성 있는 메모리 풀(Coherent Memory Pool)을 구성할 수 있도록 함으로써 동적인 메모리 관리 및 대규모 확장을 가능하게 한다 [26, 57–61]. 이러한 CXL 기술을 구현할 수 있는 연결 반도체의 설계자산(IP, Intellectual Property)과 다양한 장치를 CXL로 연결하게 하는 스위치 반도체를 통해 전통적인 메모리 확장의 제약을 극복하고 데이터 이동을 제거하여, GPU 간 자원 활용 효율을 획기적으로 향상할 수 있다.

CXL과 같은 신흥 인터커넥트 솔루션을 활용하여 현재 AI 인프라의 구조를 새롭게 설계하려면, AI 모델의 이론적 구조에서부터 데이터센터의 하드웨어 구성까지의 전체 시스템을 폭넓고 쉽게 이해하는 것이 필수적일 것이다. 본 기술 보고서에서는 먼저 현대 AI 모델이 데이터를 어떻게 표현하고 처리하는지에 대한 원리를 쉽게 정리하고, 이후 데이터 관리, 메모리 구조 및 통신 최적화 기술의 관점에서 주된 확장성 한계를 분석한다. 이 과정에서 실제로 최신 NVIDIA GPU 아키텍처들로 구현된 대규모 AI 인프라 사례를 통해, 대규모 AI 워크로드 처리에 대해 GPU 기술이 갖는 구조적 한계를 구체적이고 명확하게 살펴본다.

그후, 본 기술 보고서에서는 점점 복잡해지고 다양한 특성을 가지는 AI 워크로드를 더 잘 처리하기 위해, CXL을 기반으로 하는 모듈형(Modular) 및 동적 컴포저블(Composable) 데이터센터 구조를 제안한다. 이 구조는 연산, 메모리, 가속기와 같은 자원들을 논리적 또는 물리적으로 분리(Disaggregate)하여, 워크로드의 요구에 따라 자원을 유연하게 나누고 늘릴 수 있도록 만들어졌다. CXL 인터커넥트 기술은 모든 반도체 사이의 캐시 일관성(Cache Coherence)을 유지해, CPU의 개입 없이도 외부 메모리 장치와 데이터를 공유할 수 있게 한다. 이를 통해, 일반적인 AI 학습과 추론뿐 아니라, 검색 기반 생성(RAG, Retrieval-Augmented Generation [41, 42, 62–64])이나 키-값 캐싱(KV Caching, Key-Value Caching [29, 31, 65–68])과 같은 다양한 추가 작업까지도 새로운 패러다임의 병렬·분산 처리를 적용하여 획기적인 성능 실현과 비용 절감을 이룰 수 있다.

본 기술 보고서에서는 AI 인프라 혁신을 목표로, 수십만 대의 GPU 및 가속기들이 마치 하나의 거대한 연산장치처럼 동작할 수 있게 하는 다양한 연결 반도체 기술과 이를 위한 대규모 데이터센터 인터커넥트 솔루션을 소개한다.

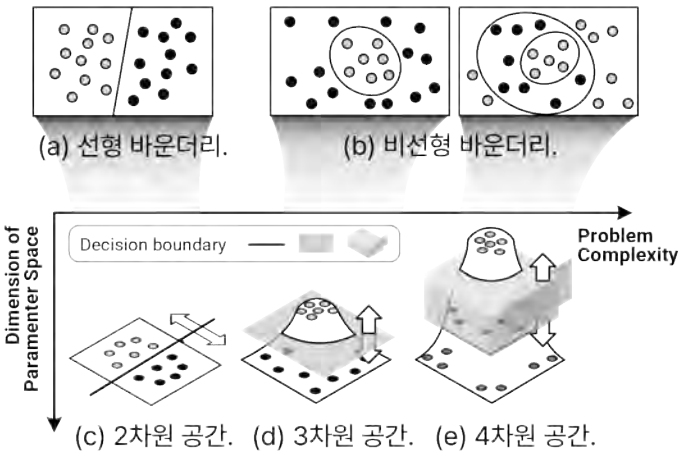
이외에도, 본 기술 보고서는 수십만 대의 가속기가 하나의 망으로 연결될 수 있도록 하기 위해, 인터커넥트 기술들을 하나로 통합하는 기술을 제안한다. 이를 위해, 고처리량·저지연 가속기 전용 인터커넥트 기술인 울트라 엑셀러레이터 링크(UALink [69, 70]), 엔브이링크(NVLink [71–73]), 엔브이링크 퓨전(NVLink Fusion [74, 75])을 통합한 엑스링크(XLink) 아키텍처를 제안한다. UALink는 개방형 이더넷(Ethernet)에 기반하며, NVLink는 엔비디아 플랫폼 전용으로 설계된 링크이다. 두 기술 모두 단일 홉(Single Hop) 연결 방식을 사용하여 랙 내의 소규모 가속기 간 고성능 분산 처리에 적합하지만, 대규모 AI 시스템의 확장성 측면에서는 한계가 있다. 이를 보완하기 위해 본 보고서는 CXL과 XLink를 융합한 "하이브리드 링크 아키텍처(CXL-over-XLink)"를 제안하여, 서로 다른 종류의 가속기 클러스터들을 통합한 슈퍼클러스터(Supercluster)를 구성하고, 메모리 공유와 노드 간 통신의 확장성을 향상한다. 이를 통해 불필요한 원격 직접 메모리 접근(RDMA, Remote Direct Memory Access) 기반 데이터 이동 [76–79]을 감소시키고, 현대적 AI 인프라에 적합한 확장 가능한 통신 구조를 제공한다.

마지막으로, 본 보고서는 제안된 하이브리드 링크 기반 슈퍼클러스터 아키텍처에 대해 로컬 가속기 메모리와 외부 메모리 풀을 계층적으로 결합하여 다양한 지연 시간과 용량 요구사항에 대응하는 메모리 풀링 구조를 제시한다. 이와 더불어 HBM, 실리콘 포토닉스, 저비용의 계층형 CXL 메모리 전략 등을 활용하여 시스템의 물리적 확장성과 비용 효율을 동시에 확보할 수 있는 방안을 검토하고, HPC 환경의 과학 계산 및 메시지 전달 인터페이스(MPI, Message-Passing Interface) 기반 워크로드 [80–84]에서도 이러한 기술이 실질적으로 확장될 수 있음을 분석하여, CXL 기반 AI 인프라의 활용성과 실용성을 입증한다.

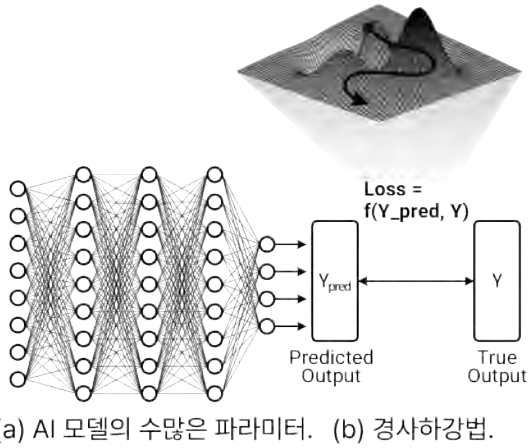
2. RNN부터 트랜스포머까지, 시퀀스 모델링의 패러다임 전환

본 절에서는 최근 AI 기술이 연구 중심 영역에서 벗어나, 일상생활과 산업 현장에 실질적으로 사용될 수 있게 된 배경에 대해서 알아보고 AI를 최대한 쉽게 설명하고자 한다. 이를 위해, 현대의 AI 모델이 복잡한 데이터를 어떻게 표현하고 처리하는지를 살펴보고, AI의 최근 급격한 변화와 성장을 가능하게 한 하드웨어 기술 발전 과정을 정리하여 소개할 것이다.

이러한 AI 소개와 설명의 중심에는 시퀀스 모델링(Sequence Modeling)의 데이터 처리 방식과 발전 과정이 자리 잡고 있다. 본 기술 보고서는 초기의 "시퀀스-투-시퀀스(Seq2Seq, Sequence-to-Sequence)" 프레임워크 [85–87]에서 출발하여, 어텐션(Attention) 메커니즘과 트랜스포머(Transformers [6, 88, 89]) 구조의 등장 및 발전을 거쳐, 궁극적으로 오늘날의 "대규모 언어모델(LLM, Large Language Models [46, 90–94])"에 이르는 흐름을 다룰 것이다. 또한, 시계열 데이터(Time-Series Data)의 특성과 중요성을 설명하고, 이를 처리하는 Seq2Seq 프레임워크의 핵심 개념을 제시한다. 또한, "순환 신경망(RNN, Recurrent Neural Network [95–97])" 기반 초기 모델의 장점과 한계점을 분석한 뒤, 어텐션 메커니즘이 이를 어떻게 보완했는지 다룰 것이다. 이어 트랜스포머 아키텍처가 병렬 연산과 하드웨어 가속 기술의 발전을 활용하여 기존 모델의 확장성(Scalability) 문제를 극복한 방식을 소개하고, 이러한 확장성 개선이 실제 응용 분야에서 트랜스포머 구조가 빠르게 확산된 배경임을 설명한다. 마지막으로 트랜스포머에서 LLM으로 발전하는 과정과, 이 과정에서 요구되는 대규모 하드웨어 인프라 및 시스템 확장성 측면의 기술적 시사점을 알아보도록 한다. 이를 통해 본 절은 현대 AI 시스템의 복잡한 인프라 설계와 운영 환경을 이해하는 데 필요한 핵심 개념을 제공할 것이다.



|그림 1| 디시전 바운더리 설정.



|그림 2| 손실함수의 최소화.

2.1. 시계열 데이터와 시퀀스-투-시퀀스(Seq2Seq) 프레임워크의 이해

최근 AI 기술은 다양한 실제 문제에 대해 우리가 도달할 수 있는 높은 수준의 정확도를 달성하며, 다양한 영역에서 폭넓은 주목과 혁신을 이끌어내고 있다. 시퀀스 모델을 설명하기에 앞서, 오랫동안 사용되어 왔던 AI 모델 학습과 추론에 대하여 그 원리를 간단히 알아보고 이러한 모델에서 시계열 데이터를 어떻게 처리할 수 있었는지를 설명한다.

AI 학습과 추론 방법에 대한 간단한 소개. 현실의 복잡한 문제를 해결하기 위해, 실제 데이터는 AI가 처리할 수 있도록 일반적으로 숫자, 벡터, 행렬과 같은 특정 차원(Dimension)의 좌표계로 사상(Mapping)할 수 있는 수학적 표현으로 변환된다. 그림 1에서 볼 수 있듯이 AI는 현실 데이터가 사상된 좌표계에서 특정 패턴을 가진 그룹들의 경계를 나누는 기준, 즉 디시전 바운더리(Decision Boundary)를 설정하여 추후 입력되는 문제가 어떤 그룹에 들어가는 것인지 확인하고 문제를 해결하는 것이 핵심이다. 이를 위해 AI 모델이 데이터 간 그룹을 설정하는 좌표계를 옮겨 다니며 범주를 탐색함으로써 서로 간 그룹을 구분하는 범위를 좀 더 명확히 하는 디시전 바운더리를 찾는 과정이 바로 모델 학습(Training)이라고 볼 수 있다. 이에 반해, 모델 추론(Inference)은 간단히 이야기하여 특정 입력에 대한 질의가 있을 때 해당 입력이 좌표계 어디에 있고 이 디시전 바운더리에 의해서 어디에 포함되는지에 대해 판단해주는 것으로, 이 추론이 실제 결과와 유사할수록 정확도(Accuracy)가 높다고 할 수 있다.

AI는 현실 데이터가 사상된 좌표계에서 특정 패턴을 가진 그룹들의 경계를 나누는 기준을 설정(학습)하여 추후 입력되는 문제가 어떤 그룹에 들어가는 것인지 확인(추론)하고 문제를 해결하는 것이 핵심이다.

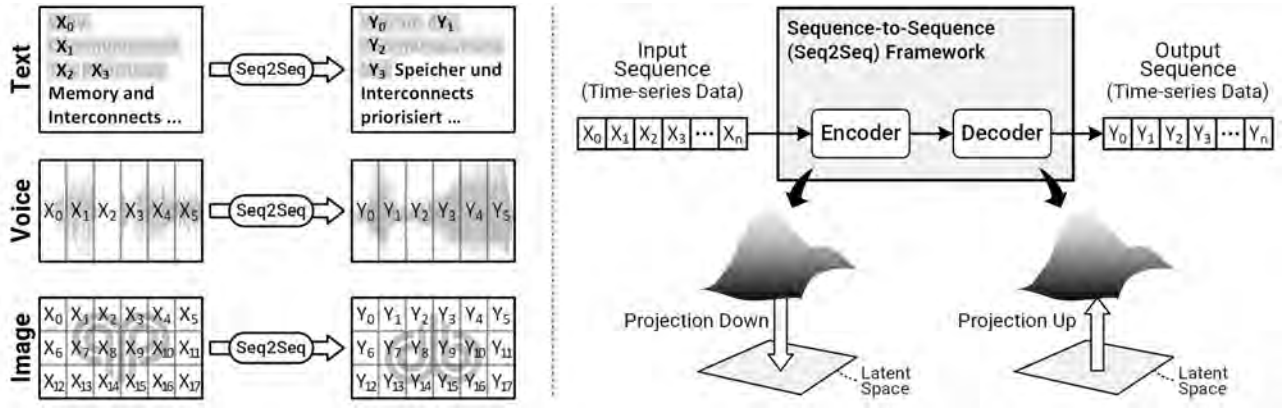
여기서 디시전 바운더리를 결정한다는 것(학습)은 실질적으로는 손실함수(Loss Function)라는 지표를 이용하여

모델의 예측 결과가 실제 정답과 얼마나 다른지를 측정하는 것을 의미한다. 디시전 바운더리는 이 손실값을 최소화하도록 설정되며, 학습을 통해 도출된 최적의 디시전 바운더리는 높은 추론 정확도를 제공하여 다양한 실제 문제들을 효과적으로 해결할 수 있다. 그러나 현실에서 접하는 문제들은 단순한 저차원 공간에서 다양한 데이터 그룹들을 나눌 수 있는 좌표상 경계를 명확하게 설정하기 어려운 경우가 훨씬 많다. 예를 들어 그림 1a처럼 2차원 좌표평면계에서 두 개의 다른 패턴을 가지는 그룹을 디시전 바운더리를 선형함수 하나로 쉽게 판단할 수 있는 경우도 있으나, 그림 1b처럼 해당 방식으로 해결되지 않는 경우가 대부분이다. 그림 1c(2차원)에서의 한계점을 극복하기 위해 그림 1d(3차원)과 그림 1e(4차원)에 나타낸 것처럼 AI 모델은 파라미터 공간의 차원을 증가시켜 더 복잡한 형태로 데이터를 표현하고, 이렇게 표현된 높은 차원의 파라미터 공간에서 데이터를 더 효과적으로 분류할 수 있는 디시전 바운더리를 찾아낸다. 예시에서는 단순한 차원만을 이야기하였지만, 현대 대규모 AI 모델이 가지는 파라미터 공간의 크기(차원)는 일반적으로 수십억(Billions)에서 수조(Trillions) 수준이다 [7, 28, 98–101]. 이러한 대규모 파라미터 공간 크기는 이제까지 풀지 못했던 문제를 해결하는 데 큰 역할을 하고 있지만, 막대한 양의 메모리와 데이터 통신을 필수적으로 필요로 하여 AI 인프라 구축에서 새로운 도전 과제로 부각하고 있다.

현실에서 접하는 문제들은 단순한 저차원 공간에서 다양한 데이터 그룹들을 나눌 수 있는 좌표상 경계를 명확하게 설정하기 어려운 경우가 훨씬 많기 때문에 현대 대규모 AI 모델은 수십억에서 수조에 이르는 차원의 파라미터를 사용한다.

참고로, 앞에서 언급된 손실함수의 대표적 예로는 평균제곱오차 손실(Mean Squared Error Loss [102–105]) 이나 교차 엔트로피 손실(Cross-Entropy Loss [106–110])등이 있으며, 그림 2에서처럼 AI 모델 학습의 목표는 기본적으로 이러한 손실함수가 최솟값을 갖도록 하는 모델 파라미터를 설정하는 것에 가깝다. 다만 파라미터의 개수가 매우 많으므로, 최적의 데이터 표현에 대한 차원이 결정된다고 하더라도 AI 모델은 해석적인(Analytic) 방법으로 손실함수의 최솟값을 찾기 어려운 경우가 다반사이다. 따라서 AI 학습에서는 근사적 알고리즘인 경사 하강법(Gradient Descent) [15, 102, 111, 112]이 널리 사용된다. 경사 하강법은 손실을 최소화하기 위해 임의의 파라미터 초기값에서 시작하여, 기울기 정보를 이용하여 함수의 값이 줄어드는 방향으로 이동하는 것을 반복하는 형태로 구현된다. 손실함수가 최솟값을 갖는 지점에서는 기울기가 0이 되므로 이 조건에서 알고리즘을 종료한다.

경사 하강법은 복잡한 손실함수에서도 효과적으로 최솟값을 찾을 수 있다는 장점이 있으나, 명확한 한계점도 존재한다. 반복 탐색을 통해 최솟값에 접근하므로, 한 번에 이동하는 거리(Step Size)가 너무 크면 최소 지점을 지나칠 수 있고, 반대로 너무 작으면 탐색 시간이 지나치게 길어질 수 있다. 또한, 임의의 지점에서 시작하여 손실을 줄이는 방향으로 이동하므로 전역 최솟값(Global Minimum)이 아닌 지역 최솟값(Local Minimum)에 갇힐 가능성도 있다. 이러한 한계를 극복하기 위해 확률적(Stochastic) 경사 하강법 [5, 102, 113–118])이나 Adam(Adaptive Moment Estimation [119–123])과 같이 Step Size나 이동 방향을 동적으로 조정하는 다양한 최적화 기법들이 제안되었으나, 결국에는 손실함수가 최소값을 갖도록 모델 파라미터를 설정하는 AI 학습 목표를 동일하게 가지고 있다.



(a) 시계열 데이터 시퀀스. **(b)** Seq2Seq 모델의 핵심 개념.

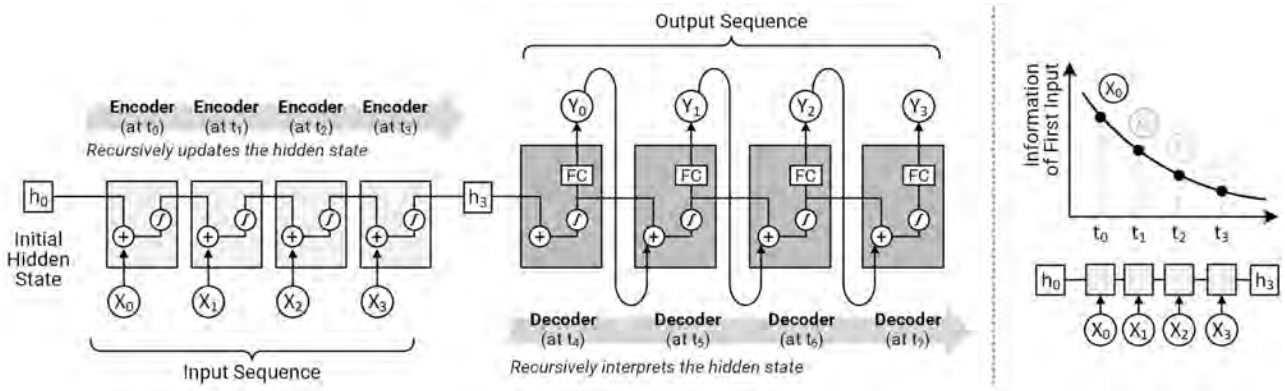
|그림 3| 시퀀스-투-시퀀스(Seq2Seq) 프레임워크.

시퀀스 모델링의 기본 구성요소. 현실 세계의 문제들은 이미지, 오디오, 비디오, 텍스트 등 여러 형태의 데이터 모달리티(Modality)를 포함하지만, 공통적으로 시간에 따른 "순차적 특성"을 지닌다. 이 시계열적 속성은 다양한 데이터 유형을 일반화된 "시계열 데이터"로 표현하고 분석할 수 있는 기반을 제공하며, 이를 쉽게 이해할 수 있도록 그림 3a에 시각적으로 표현하였다. 그림에서 보여지듯, 다양한 형태의 데이터와 문제는 시계열 데이터로 표현될 수 있다. 따라서 시계열 데이터에 내재된 시간적 종속성과 구조를 효과적으로 포착하는 것은 AI 시스템이 깊이 있는 분석을 통해 정교한 의사결정을 수행하고, 다양한 응용 분야에서 높은 예측 정확도를 달성하는 데 핵심적인 역할을 한다고 볼 수 있다. 하지만 앞서 언급된 것처럼 AI 시스템에 입력되는 이러한 데이터들은 구조화된 수치 형태로 먼저 변환되어야 한다. 수치 변환을 위해 제안된 시퀀스 모델 중 가장 영향력이 있는 모델이 Seq2Seq이다. 2014년에 처음 소개된 이 모델 [85]은 비교적 오래된 접근 방식이지만, 그 핵심 구조는 오늘날 다양한 시퀀스 모델링 기법의 근간으로 다른 모델들을 이해하는데 매우 중요하기 때문에 본 절에서는 이를 간단히 정리하여 설명하고자 한다. 관련하여 Seq2Seq를 이해하는데 핵심이 되는 인코딩(Encoding), 순서화(Ordering), 디코딩(Decoding)이라는 세 가지 개념을 그림 3b를 통해 구조적으로 도식화하고 아래 설명하였다.

시계열 데이터의 시간적 특성과 구조를 잘 포착하는 것이 AI 시스템의 정교한 의사결정과 높은 예측 정확도 달성에 핵심이다. 수치화를 위한 시퀀스 모델의 구조는 기본적으로 순서화, 인코딩, 디코딩이라는 개념들을 반드시 포함한다.

---

1. **인코딩:** 인코더의 주요 작업은 입력 시퀀스를 압축하여 간결한 수치 표현인 잠재 벡터(Latent Vector)로 변환하는 것이다. 이 벡터는 입력 데이터에 존재하는 핵심적인 맥락 정보와 시간적 관계를 효과적으로 내포하며, 이후 시퀀스



(a) 인코딩 과정. **(b)** 디코딩 과정. **(c)** 기울기 소실 문제.

|그림 4| RNN 기반 Seq2Seq 아키텍처와 그 한계점.

분석 및 예측의 기초로 사용된다.

2. **순서화:** 순차적 처리 과정은 모델 내부의 각 단계가 이전 요소의 맥락 정보를 충분히 반영할 수 있도록 한다. 이러한 순서 정보는 출력 결과의 일관성과 정확성을 보장하며, 일반적으로 시계열 예측의 신뢰도를 높이는 데 중요한 역할을 한다.
3. **디코딩:** 디코더는 인코더가 생성한 내부 표현을 기반으로 사람이 이해할 수 있는 출력 시퀀스를 재구성한다. 이 과정에서는 잠재 벡터가 지닌 문맥 정보를 활용하여 번역, 요약, 예측과 같은 실제 결과물을 생성한다.

이러한 특성을 반영한 Seq2Seq 모델의 인코더-디코더 프레임워크는 길이와 복잡성이 다양한 시퀀스를 효율적으로 처리할 수 있도록 구조화 되어 있다. Seq2Seq의 인코더는 전체 입력 시퀀스를 간결한 내부 표현으로 압축하면서 문맥적(Contextual) 정보 및 시간적(Temporal) 정보를 효과적으로 보존할 수 있도록 한다. 디코더는 이렇게 생성된 내부 표현을 바탕으로 정확하고 일관된 출력 시퀀스를 생성하여, 복잡한 데이터 패턴과 인간이 이해할 수 있는 출력 사이의 격차를 줄이고, 효과적으로 연결하는 역할을 수행한다. 이와 같은 구조화된 방법론 덕분에 Seq2Seq 모델은 언어 번역, 음성 인식 등 다양한 작업에서 효과적인 도구로 자리 잡았다.

그림 4에 나타난 것과 같이, 초기 Seq2Seq 모델은 순차적 데이터를 자연스럽게 처리할 수 있는 RNN을 기반으로 구현되었다. RNN 기반의 Seq2Seq 아키텍처에서 인코더의 주된 역할은 입력 시퀀스를 읽고 이해하는 것으로 입력 데이터를 순차적으로 처리하며(그림 4a), 이전 데이터의 핵심 문맥 정보를 요약한 '은닉 상태(Hidden State)'라는 내부 메모리를 유지하도록 만들어졌다. 인코더는 단계마다 현재 데이터와 이전 은닉 상태를 비선형 함수(Non-linear Function, 예: 하이퍼볼릭 탄젠트(tanh), 정류 선형 유닛(ReLU, Rectified Linear Unit [124, 125]))를 포함한 수학적 연산을 통해 결합하여 새로운 은닉 상태를 생성함으로써 순차적으로 데이터를 만드는 것이 중요한 작업 중 하나이다. 이러한 비선형 함수는 데이터 내의 복잡하고 비선형적인 관계를 효과적으로 포착할 수 있도록 돕는 역할을 수행하는데, 이와 같은 비선형 변환이 없다면 모델은 복잡한 패턴과 시퀀스 내 시간적 종속성(Dependency)을 표현하는 데 심각한 제한이 생긴다 [126–128]. 지금까지 설명한 방식으로 각 단계에서 현재 입력 데이터와 이전 시점의 은닉 상태를 비선형적으로 결합하여 순서대로 처리하는 업데이트 과정은, RNN 기반의 Seq2Seq 모델의

인코더가 입력 시퀀스의 시간적 종속성과 문맥 정보를 보존하면서 전체 시퀀스를 점진적으로 응축된(Concise) 형태의 의미 있는 내부 표현으로 요약할 수 있게 도와준다.

그림 4b에 나타난 것처럼, 인코더가 시퀀스 압축을 완료하면 RNN 기반의 Seq2Seq 아키텍처에서 디코더는 이 요약된 표현을 바탕으로 문맥을 이해할 수 있고 이를 통해 구조화된 출력 시퀀스를 재구성한다. 해당 디코더는 압축된 은닉 상태뿐만 아니라 이전에 생성한 출력을 입력으로 활용하여 출력 데이터를 순차적으로 생성하는데, 재구성 과정에서 디코더는 완전 연결(FC, Fully Connected) 레이어(Layer)를 사용하도록 설계되어있다 [88, 129]. FC 레이어는 추상적이며 압축된 내부 표현을 실제 이해 가능한 현실 세계의 출력 형태로 변환하는 데 특화된 신경망 계층이다. 각 FC 계층은 내부 정보를 더욱 명료하고 구체적인 형태로 변형하여, 디코더가 정확하고 일관된 번역, 요약, 예측과 같은 시퀀스를 생성할 수 있도록 돕는다.

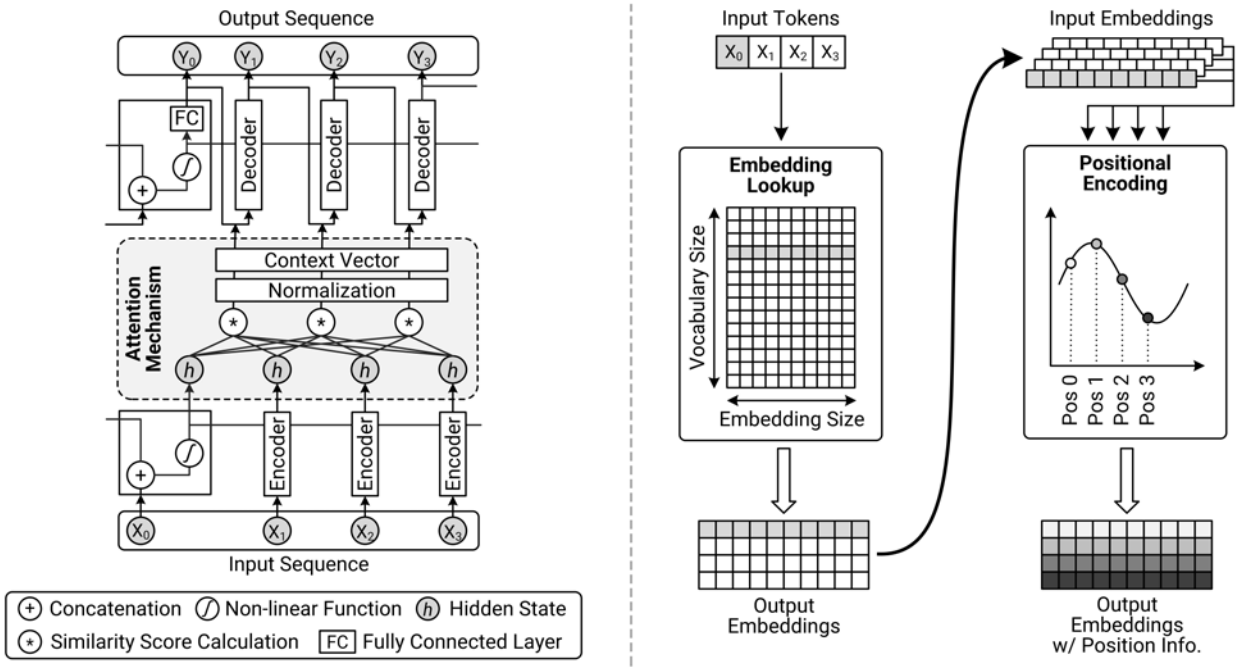
초기 시퀀스 모델링은 초반부 데이터에서 나온 정보가 점차 없어지는 기울기 소실 문제, 이른바 망각 현상과, 순차적인 연산 구조로 인한 종속성 문제로 병렬화 되지 못해 확장성과 계산 효율성이 제한되는 문제를 가지고 있다.

RNN 기반의 Seq2Seq 모델은 이러한 초기 성공에도 불구하고 그림 4c에서 설명하는 바와 같이 훈련 중 시퀀스의 초반부 데이터에서 나온 정보가 점차 소실되는 "기울기 소실 문제(Vanishing Gradient Problem [130, 131])"로 인해 그 사용이 크게 제약되었다. 기울기 소실 문제로 인해 모델은 긴 범위 데이터 간의 연관성(Long-range Dependency)을 포착하는 능력이 현저히 저하되는데, 이 문제는 인간이 오래된 기억을 점차 잊어버리는 현상, 즉 망각과 유사하다고 볼 수 있다. 이러한 기울기 소실 문제 이외에도, RNN은 본질적으로 순차적인 연산 구조를 가지기 때문에 병렬 연산 기술의 혜택을 받지 못하며, 이에 따라 확장성과 계산 효율성이 제한되는 문제를 가지고 있다. 이러한 RNN 기반의 Seq2Seq 모델 한계점들은 장거리 문맥 정보를 더 잘 보존하고 병렬 연산이 가능한 더 진보된 아키텍처가 개발되는 계기가 되었다.

2.2. 시퀀스 모델링의 패러다임 전환

어텐션 메커니즘의 도입. 전통적인 RNN 기반 아키텍처는 구조적으로 기울기 소실 문제와 제한된 확장성을 내재하고 있다는 것을 알아보았다. 본 절에서는 이 한계점 중 망각 문제를 해결하기 위한 시퀀스 모델링 기술의 핵심 전환점으로 평가받는 어텐션(Attention) 메커니즘을 구체적으로 설명하고자 한다 [88, 132, 133].

어텐션 메커니즘은 모델이 출력 시퀀스의 각 요소를 생성할 때 입력 시퀀스 내에서 가장 관련성 높은 부분에 동적으로 집중하여, 초기 정보를 유지하고 기울기 값을 최대한 보존할 수 있도록 하는 기술이다. 다시 말해, 기존의 RNN 기반



(a) RNN에 적용된 어텐션 메커니즘. **(b)** 임베딩 레이어. **(c)** 위치 인코딩.

|그림 5| 어텐션 기반 RNN 및 트랜스포머 아키텍처.

접근법은 전체 입력 시퀀스를 하나의 고정된 크기의 은닉 상태로 압축하여 표현하는 방식이기 때문에 훈련 중 기울기 값들을 잃어버리고 망각하는 형태의 문제를 만들어냈다. 반면, 어텐션 메커니즘은 인코더가 생성한 모든 은닉 상태에 대한 접근을 유지하고, 현재 디코딩 단계와의 관련성에 따라 각 상태에 가중치를 선택적으로 부여한다. 이 과정에서 앞부분에 위치한 입력이라도 높은 관련성을 가지면 충분한 영향력을 가질 수 있기 때문에, 시퀀스의 길이와 관계없이 정보가 소실되지 않고 유지될 수 있다.

그림 5a는 이러한 어텐션 메커니즘을 좀 더 구체적으로 설명해준다. 먼저, 디코더는 순서상 각 단계에서 자신의 현재 은닉 상태와 모든 인코더 은닉 상태 간의 유사도를 계산하여 어텐션 가중치(Attention Weights)를 생성한다. 이후, 소프트맥스(Softmax) 함수 [134–136] 등을 통해 이 유사도 점수를 정규화하고 디코더는 이 가중치를 이용해 인코더의 은닉 상태들을 가중 합산하여, 현재 단계에서 가장 관련 있는 입력 정보를 담은 컨텍스트 벡터(Context Vector)를 생성한다. 생성된 컨텍스트 벡터는 이전 출력들과 함께 사용되어 다음 출력 토큰(Token[3])을 생성하는 데 활용된다. 즉, 컨텍스트 벡터는 중요한 정보를 선택적으로 집중해서 반영할 수 있으므로, 시퀀스가 길어져도 기울기 정보가 희석되지 않고 유지될 수 있는 구조적 이점을 제공하며, 결과적으로 기울기 소실(망각 현상)에 관련된 문제를 완화한다.

우리는 이러한 어텐션 메커니즘 도입과 관련하여 두 가지 특성을 이해하는 것이 필요하다. 첫째, 어텐션 기반 모델은 입력 시퀀스의 길이와 관계없이 어떤 위치에 있는 정보라도 효과적으로 유지하고 참조할 수 있다. 이는 앞서 언급한 것처럼 고정 크기의 은닉 상태로 모든 정보를 압축했던 기존 방식의 한계를 극복하며, 필요한 정보를 상황에 따라 동적으로 활용하는 콘텐츠 기반 메모리 구조(Content-Based Memory)를 가능하게 한다 [88, 132, 133, 137].

[3]토큰은 단어, 서브워드, 문자와 같은 언어 데이터의 기본 구성 단위를 의미한다.

둘째, 어텐션 메커니즘은 모델의 해석 가능성(Interpretability)을 크게 향상시킨다. 다시 말해 생성된 어텐션 가중치는 모델이 입력 시퀀스 중 어느 부분에 집중하여 출력을 생성했는지를 시각적으로 보여줄 수 있으며, 이는 복잡한 모델의 의사결정 과정을 직관적으로 이해하는 데 큰 도움이 된다 [138–140].

어텐션 메커니즘은 초기 정보를 유지하고 기울기 값을 최대한 보존할 수 있도록 하는 기술로 망각 현상을 완화시키는 핵심이 되었으나 여전히 순차적 계산 방식으로 병렬처리가 불가능하여 본질적으로 성능이 제한되는 문제를 가지고 있다.

이러한 개선에도 불구하고, 어텐션 기반 모델은 여전히 기존 RNN 아키텍처의 순차적 계산 방식을 계승하기 때문에 본질적인 병목(병렬성의 부재)을 해소하지는 못한다. 각 시간 단계는 이전 단계의 계산 결과에 반드시 의존하므로, 현대의 병렬 하드웨어를 충분히 활용하기 어렵다. 이러한 병목은 완전한 어텐션 기반 구조로 설계된 트랜스포머(Transformer)의 등장을 촉진했으며, 결과적으로 시퀀스 모델링의 구조적 패러다임을 근본적으로 변화시켰다.

순환에서 병렬로: 트랜스포머(Transformer)의 혁신. 시퀀스 모델링 분야의 중대한 전환점은 2017년 트랜스포머(Transformer) 아키텍처 [88]의 도입으로 시작되었다. 트랜스포머는 기존 Seq2Seq 및 RNN 기반 모델의 순차적 연산 구조를 벗어나 오직 어텐션 메커니즘만으로 완전한 병렬 처리를 실현함으로써, 시퀀스 모델의 구조적 패러다임과 계산 효율성을 근본적으로 변화시켰다.

트랜스포머의 핵심 연산 구조는 셀프어텐션(Self-Attention) 기법에 있다 [141–145]. 기존의 어텐션 메커니즘이 주로 인코더와 디코더 간의 관계에 중점을 두었다면, 셀프어텐션은 같은 시퀀스 내부의 토큰 간 관계를 병렬적으로 처리할 수 있도록 설계되었다. 구체적으로 셀프어텐션은 시퀀스 내 모든 토큰 간 상호작용을 동시에 계산할 수 있게 함으로써 기존 RNN 기반 어텐션이 가진 순차적 계산의 한계를 극복하고 독립적인 병렬 처리를 가능하게 한다.

이러한 병렬 처리가 가능한 이유는 셀프어텐션 연산이 각 토큰의 "입력 임베딩(Input Embeddings)"표현에 기반하여 동시에 이루어지기 때문이다. 입력 시퀀스의 각 토큰은 임베딩 레이어를 통해 고차원 벡터로 변환되고, 그림 5b에서 보는 바와 같은 임베딩들이 모델 입력으로 활용된다. 트랜스포머는 RNN 기반 모델과 달리 이 임베딩 표현들을 직접 병렬로 처리하여 시퀀스 전체를 동시에 연산할 수 있다.

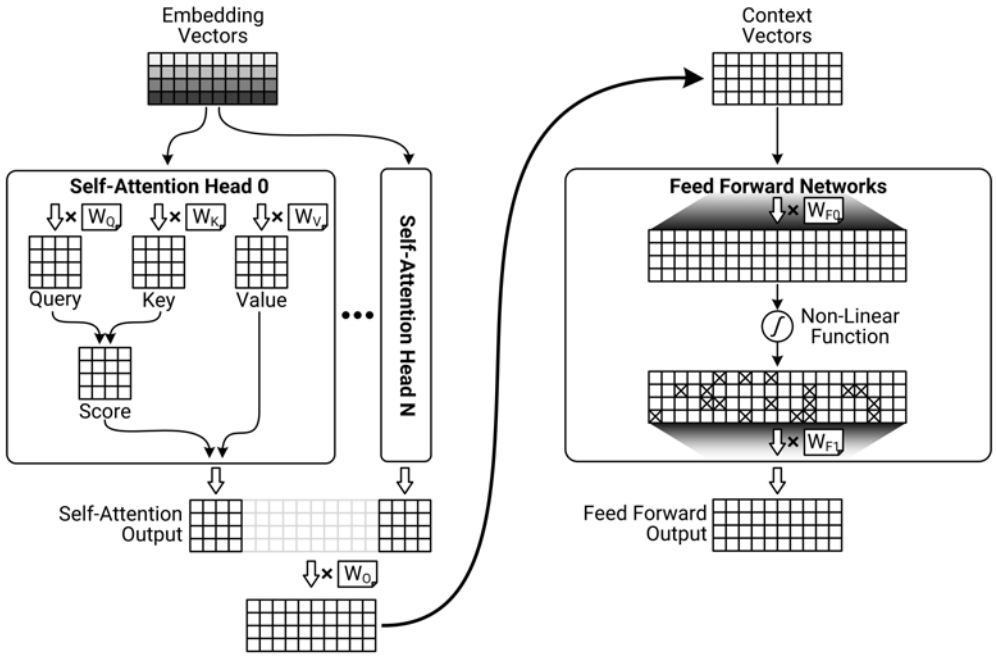
각 토큰의 임베딩은 고유한 의미적 정보를 포함하고 있으며, 셀프어텐션 메커니즘은 개별적 임베딩을 독립 참조함으로서 모든 토큰 간 관계를 동시에 반영하여 처리할 수 있다. 이로써 순차적 모델링이 필요했던 기존 RNN의 제약이 사라지고, 모든 토큰 간의 계산을 병렬로 수행할 수 있는 기반이 마련되었다고 볼 수 있다. 무엇보다 트랜스포머의 이러한 동시 처리 방식은 병렬 하드웨어의 성능을 극대화하여 계산 효율성과 확장성을 높일 수 있기 때문에, GPU 사용이 AI 인프라에서 대중화되는 데 중요한 역할을 하였다 [43, 146, 147].

셀프어텐션의 임베딩 기반 처리 방식은 기존 어텐션 기본 모델의 순차적 구조를 제거함으로써 병렬성을 확보할 수 있게 되었지만, 이로 인해서 입력 시퀀스의 순서 정보(Ordering)를 명시적으로 유지하지 못하는 문제를 해결해야 했다. 다시 말해, RNN은 계산 과정 자체가 시간 흐름을 내재적으로 반영하지만, 트랜스포머는 병렬 연산 구조로 인해

시간 정보를 자동으로 보존하지 못한다. 이를 해결하기 위해 위치 인코딩(Positional Encoding)이 도입되었다. 그림 5c에 나와 있듯, 위치 인코딩은 각 토큰의 상대적 또는 절대적 위치 정보를 임베딩 벡터에 추가하여 모델이 토큰의 위치를 파악할 수 있도록 해준다.

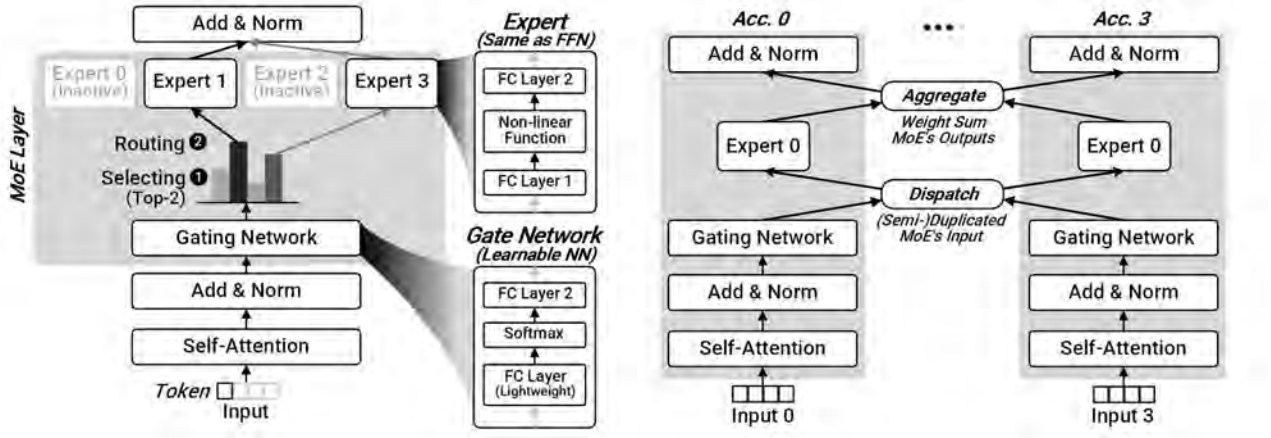
트랜스포머의 동시 처리 방식은 병렬 하드웨어의 효율성과 확장성을 높여, AI 인프라에서 GPU의 대중화에 기여하였다.

이러한 위치 인코딩은 사실 간단한 정현 함수 기반의 고정형(Sinusoidal Positional Encoding) 또는 학습 가능한 임베딩(Learned Positional Embedding)의 형태로 구현될 수 있다 [148–150]. 두 방식 모두, 실제 구현에서는 트랜스포머가 다루는 임베딩 벡터에 더해져 셀프어텐션 연산 이전에 시퀀스 내 위치 정보를 효과적으로 반영하면서도 병렬 처리의 효율성을 유지할 수 있도록 돕는다. 따라서, 트랜스포머는 임베딩 기반의 입력 표현과 위치 인코딩을 결합하여 병렬 연산을 유지하면서도 긴 거리의 의존성(Long-Range Dependencies)을 효과적으로 처리할 수 있게 되었다. 이러한 구조적 특성 덕분에 트랜스포머는 기존 RNN 기반 모델보다 성능과 확장성 측면에서 완전히 다른 우수성을 입증하며, 현재 대부분의 대규모 언어모델 및 시퀀스 처리 분야에서 사실상의 표준으로 자리 잡게 되었다.



(a) 멀티-헤드 셀프어텐션 메커니즘. **(b)** 피드포워드 네트워크(FFN).

|그림 6| 트랜스포머 레이어 연산: 셀프어텐션과 피드포워드 네트워크(FFN).



(a) MoE의 전체 구조. **(b)** MoE의 통신 복잡성.

|그림 7| 전문가 혼합(MoE) 아키텍처.

셀프어텐션과 전문가 혼합(Mixture of Experts): 시퀀스 이해 능력의 향상. 본 하위 절에서는 시퀀스 모델 능력 향상의 핵심이 되는 셀프어텐션과 "전문가 혼합(MoE, Mixture of Experts)"에 대해서 좀 더 자세하게 이야기해보자.

앞서 간단히 설명된 것처럼 셀프어텐션은 트랜스포머 구조의 핵심 요소로, 기존 어텐션 메커니즘과 달리 동일 시퀀스 내부에서 서로 다른 위치에 있는 토큰 간의 관계를 병렬적으로 처리, 계산할 수 있도록 설계되었다. 이를 위해서 그림 6a에서 나타난 바와 같이, 셀프어텐션은 각 토큰의 임베딩에서 쿼리(Query), 키(Key), 밸류(Value)의 세 가지 벡터를 생성한다. 이 벡터들은 각각 학습 가능한 선형 변환 행렬(W^Q, W^K, W^V)을 통해 계산되는데 여기서 쿼리는 각 토큰이 원하는 정보, 키는 각 토큰이 제공 가능한 정보, 밸류는 실제 제공하는 정보의 내용을 나타낸다. 모든 쿼리 벡터는 전체 키 벡터들과 스케일 조정된 내적(Scaled Dot-Product)을 통해 유사도 점수(Attention Score)를 계산하며, 소프트맥스 함수를 통해 정규화된 어텐션 가중치를 산출한다. 이러한 가중치들을 밸류 벡터에 가중합 형태로 적용하여 각 토큰의 컨텍스트 벡터를 만들어 각 토큰을 시퀀스 내 다른 위치와의 의미적 관련성을 반영하는 표현으로 재구성한다. 이때 모든 토큰의 쿼리, 키, 밸류 벡터는 서로 독립적으로 생성되고, 어텐션 가중치의 계산 또한 모든 토큰 쌍에 대하여 동시다발적으로 수행될 수 있기 때문에, 병렬 연산이 가능한 GPU와 같은 하드웨어에서 효율적으로 구현될 수 있다.

트랜스포머는 나아가 셀프어텐션을 확장하고 병렬성을 최대화하기 위해 "멀티 헤드 어텐션(Multi-Head Attention [88, 151, 152])"이나 "쿼리 그룹 어텐션(Grouped Query Attention" [153–155]) 사용한다. 이는 여러 개의 셀프어텐션 블록(헤드)을 병렬로 구성하여 각 헤드가 독립적인 Q, K, V 벡터를 생성하고 연산하도록 하도록 하는 구조이다. 각 헤드는 별도로 학습된 파라미터를 통해 더욱 풍부한 의미적 표현을 얻을 수 있으며, 트랜스포머는 멀티 헤드 어텐션의 병렬 구조를 통해 다양한 시멘틱 관계 및 장기 의존성을 동시에 포착할 수 있다.

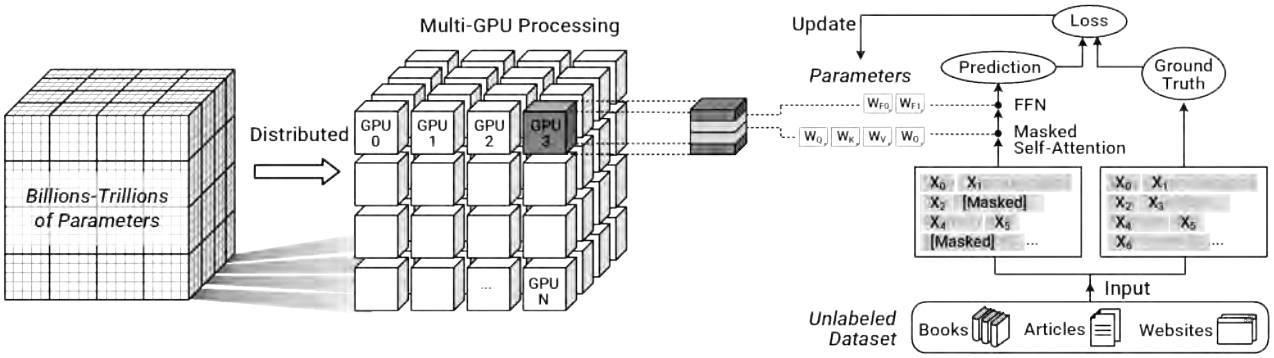
셀프어텐션에 덧붙여, 그림 6b에서 볼 수 있듯이, 트랜스포머는 일반적인 신경망에서 흔히 언급되는 FC 레이어와는 다른 특수한 구조의 "피드포워드 네트워크(FFN, Feed-Forward Network)"를 내부에 포함하고 있다. 셀프어텐션이 토큰 간 맥락적 관계를 효과적으로 포착하지만, 개별 토큰의 표현은 추가적인 정제 과정이 필요하다. 이를 위해

FFN은 셀프어텐션 메커니즘의 보완 역할로 각 토큰 임베딩에 독립적이고 위치별(Position-Wise)로 비선형 변환을 수행한다. 구체적으로, FFN은 두 개의 선형 변환과 이 사이에 ReLU나 GELU [124, 156]와 같은 비선형 활성화 함수를 결합한 구조로 구성된다. 첫 번째 선형 변환은 입력 임베딩을 보다 높은 차원의 표현 공간으로 투영하여, 모델이 데이터 내의 복잡하고 비선형적인 패턴을 포착할 수 있도록 한다. 이후 두 번째 선형 레이어는 이렇게 정제된 표현을 원래의 차원으로 다시 투영한다. 결과적으로 FFN은 셀프어텐션이 생성한 맥락 인식 임베딩을 한층 더 강화하는 역할을 하게 된다. 또한 각 토큰 위치에서 FFN 레이어가 독립적으로 동작하기 때문에, 트랜스포머 아키텍처의 본질적인 병렬 처리 장점을 그대로 유지할 수 있다.

반면, 셀프어텐션으로 생성된 토큰 단위의 표현을 FFN이 개선하더라도, 다양하고 복잡한 데이터를 더욱 효과적으로 처리하려면 모델 용량 및 연산 효율성을 증가시켜야 한다. 따라서 트랜스포머는 진보된 아키텍처인 MoE 구조를 도입하였다 [40, 157–159]. 그림 7a는 MoE의 전반적인 구조를 나타낸 것으로, 이는 앞서 언급한 모델의 용량과 계산 효율성 향상을 목표로 설계되었다. MoE는 네트워크를 다수의 전문가(Expert) 서브 네트워크로 분할하고 각 입력 토큰마다 특정 전문가만 활성화하여 연산을 수행한다. 이러한 MoE 구조에서 각 전문가는 특정 데이터 패턴이나 특징에 특화되어 있으며, 게이팅 네트워크(Gating Network)가 입력 토큰에 따라 전문가를 선택하고 라우팅하는 방식으로 동작하게 된다. 쉬운 예로, 100개의 전문가 중 입력 토큰마다 상위 2개의 전문가만 활성화하는 방식을 들 수 있다. 이는 전체 모델의 파라미터 수를 크게 증가시키면서도 실제 추론 시 일부 파라미터만 활성화하여 계산 효율성을 유지할 수 있게 하여 모델의 성능을 최대화할 수 있다 [40, 158, 160–162].

전문가 혼합 구조의 효과적 구현을 위해 GPU 간 직접 연결이나 고대역폭 스위치 기반의 고속 인터커넥트 구조가 필수적이다.

이렇게 MoE 구조는 수용할 수 있는 계산 효율성을 높일 수 있지만, 아쉽게도 GPU 간 통신 복잡성은 증가시킨다. 그림 7b는 이러한 통신 복잡성을 도식화하고 있다. 그림에서 볼 수 있듯이, 다수의 GPU에 분산된 전문가들의 연산 결과를 취합(Aggregation)하여 후속 계층으로 전달해야 하므로, 전문가 간의 빈번한 통신이 요구된다. 특히, 다음 셀프어텐션 계층은 MoE 출력이 완전히 통합된 이후에야 연산을 시작할 수 있어 계산 병렬성이 매우 제한되는 구조적 의존성을 발생시킨다. 따라서 MoE 구조를 효과적으로 구현하기 위해서는, GPU 간 직접 연결이나 고대역폭 스위치 기반의 통신 구조가 병렬 연산 성능을 유지하는 데 중요한 역할을 하며, 새로운 AI 인프라에서는 고속의 전문가 간 통신을 보장하는 인터커넥트 구조가 필수적이다.



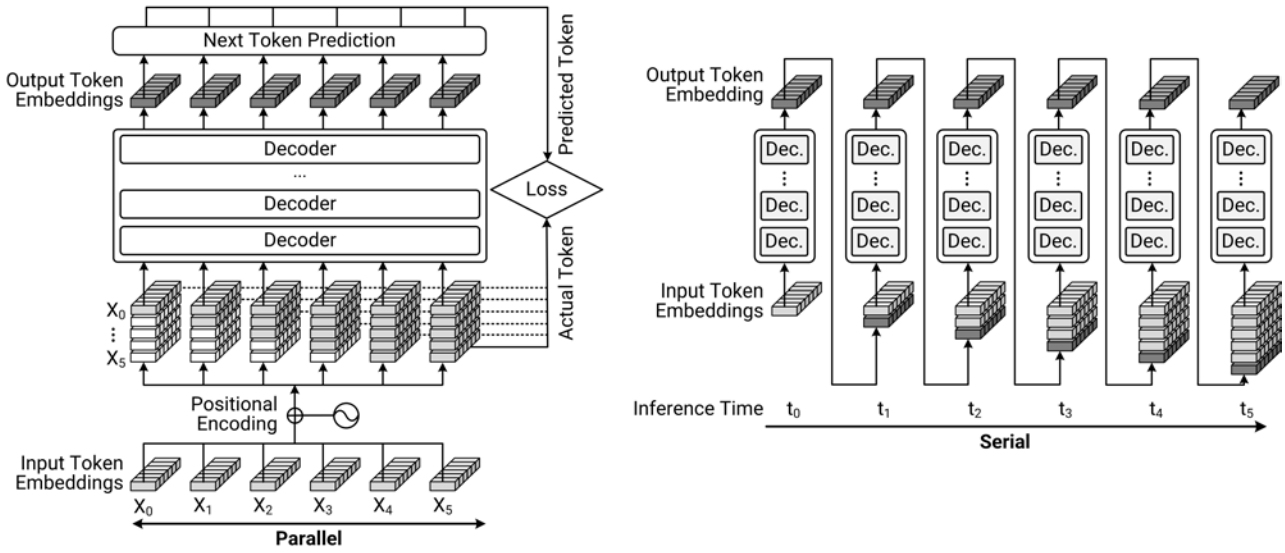
|그림 8| 병렬 처리를 활용한 LLM 사전 학습.

2.3. 트랜스포머에서 대규모 언어모델(LLM)로의 발전

앞서 분석한 것처럼, 트랜스포머의 등장은 시퀀스 모델링 분야에 큰 변화를 가져왔는데, 그 변화에서 중요한 핵심 중 하나가 병렬 처리에 있다고 볼 수 있다. 기존의 RNN은 순차적으로 연산을 수행해야하기 때문에 의존성으로 인해 병렬화가 불가능하였지만, 트랜스포머는 셀프어텐션 메커니즘을 도입해 모든 토큰 간의 관계를 한 번에 병렬로 계산할 수 있게 했다. 이로써 AI 연산 속도와 확장성이 크게 향상되었고, 자연스럽게 더 깊고 복잡한 신경망 아키텍처 개발이 가능해졌다. 이러한 트랜스포머 구조는 병렬 하드웨어, 특히 GPU와 같은 가속기의 활용을 극대화하면서, AI 인프라 하드웨어 설계에도 중대한 변화를 가져오게 했다.

대규모 학습 및 파라미터 최적화. 트랜스포머는 데이터 처리 연산을 구조적 접근 방식으로 정의하는 신경망 '아키텍처'인 것에 비해 LLM은 트랜스포머와 같은 기반 아키텍처 위에 방대한 데이터셋을 활용한 사전 학습(Pre-Training)을 수행함으로써 만들어진 구체적인 AI '모델'이다. 이러한 LLM은 트랜스포머 아키텍처의 뛰어난 확장성 덕분에 파라미터(Parameter)의 수를 크게 늘릴 수 있으며, 다양한 최적화 기법을 적용하여 지속적으로 성능을 향상시켜왔다. 대표적인 LLM의 사례로는 GPT-3, GPT-4, PaLM 등이 있으며, 이들 모두 트랜스포머 구조를 채택하여 대규모 데이터로부터 학습된 모델이라고 보면 된다. 최신 LLM 모델인 GPT-4 Turbo [90, 92, 163]와 구글(Google)의 제미니(Gemini [93, 94, 164, 165]) 등은 수십억에서 수조에 달하는 방대한 파라미터를 보유하고 있다. 여기서 파라미터는 신경망 계층 내부에서 데이터를 처리할 때 활용되는 일종의 설정값으로, 사전 학습 과정에서 광범위한 텍스트 데이터(서적, 학술 논문, 웹페이지, 소셜 미디어 등)를 기반으로 최적화된다 [6, 166, 167].

트랜스포머의 병렬 구조는 대규모 파라미터와 데이터셋을 다수의 GPU에서 동시에 처리할 수 있도록 설계되었지만, 연산 요구량이 높아 수만에서 수십만 개의 GPU를 사용해도 학습에 수주에서 수개월이 소요된다.



(a) 토큰 예측 결과와 입력 토큰의 비교를 통한 학습 과정. **(b)** 생성한 토큰을 다음 예측에 활용하는 추론 과정.

|그림 9| 오토레그레시브 모델의 동작 방식.

사전 학습은 일반적으로 셀프슈퍼바이즈드 러닝(Self-Supervised Learning [168, 169]) 기법을 많이 활용하는데 이 기법은 별도의 레이블(Label) 없이도 모델이 유의미한 표현을 자동으로 학습하도록 하는 방식이다. 대표적으로 입력 문장의 일부 단어나 문장을 마스킹(Masking)한 후, 주변 문맥을 기반으로 이를 예측하는 방법이 널리 사용된다. 그림 8에서 나타난 바와 같이, 이러한 대규모 사전 학습은 모델이 언어적 지식, 문맥 이해력, 통사적·의미적 뉘앙스뿐 아니라 일반 상식과 세계 지식까지 내부 파라미터에 내재화하도록 돕는데 이 지식은 셀프어텐션 계층의 쿼리, 키, 밸류 벡터와 FFN의 가중치 및 편향 등의 형태로 저장된다.

이러한 LLM의 학습 과정은 매우 높은 연산량과 뛰어난 병렬 처리 능력을 요구한다 [28, 170]. 특히 혼합 정밀도 연산(Mixed-Precision Arithmetic), 다중 노드 간 분산 학습, 복잡한 그래디언트 동기화 등 최적화 기법이 사용될 경우, 병렬 처리에 대한 요구 사항은 더욱 높아진다. 트랜스포머의 병렬 구조는 이러한 대규모 파라미터와 데이터셋을 다수의 GPU 클러스터에서 동시에 처리할 수 있도록 설계되었음에도 불구하고, 높은 연산 요구사항에 의해 수만에서 수십만 개의 GPU가 필요하며 이 수많은 GPU를 활용해도 수주에서 수개월 동안 지속적으로 학습이 이루어지는 것이 일반적이다 [171–173]. 따라서 이러한 대규모 시스템에서 GPU 간 빈번한 데이터 동기화와 높은 메모리 요구로 인해 분산 장치 간 확장성, 메모리 용량, 인터커넥트 설계 등 핵심 인프라 설계가 매우 중요한 이슈로 주목받고 있다.

일관성 유지와 일반화 능력 확보. 대부분의 LLM, 특히 트랜스포머 기반 모델은 학습 및 추론 단계에서 오토레그레시브(Auto-Regressive) 방식을 사용한다 [14, 167, 174]. 이는 오토레그레시브 구조가 각 토큰을 예측할 때 이전에 생성된 토큰만 참조하여 순차적으로 다음 토큰을 생성하며, 언어 데이터가 가지는 본질적인 순차적 특성을 잘 반영하기 때문이다. 예를 들어 문장 생성 시 현재의 단어는 이전 문맥만을 기반으로 결정하는데 이후의 단어들은 예측 과정에 반영되지 않기 때문에 오토레그레시브 방식은 미래 정보가 없는 상황에서도 문맥과 흐름을 유지하며 의미상 일관성 있는 텍스트를 생성하는 데 적합하다.

이러한 방식의 학습 과정을 그림 9a에서 살펴볼 수 있다. 모델은 주어진 시퀀스의 초기 토큰들만을 바탕으로 다음

토큰을 예측하며, 이때 문법적, 구문적, 논리적 흐름의 패턴을 효과적으로 학습한다. 오토레그레시브 학습 방식은 이 과정에서 시계열적 종속성을 습득하게 되어 생성된 텍스트의 일관성을 보장할 수 있게 된다.

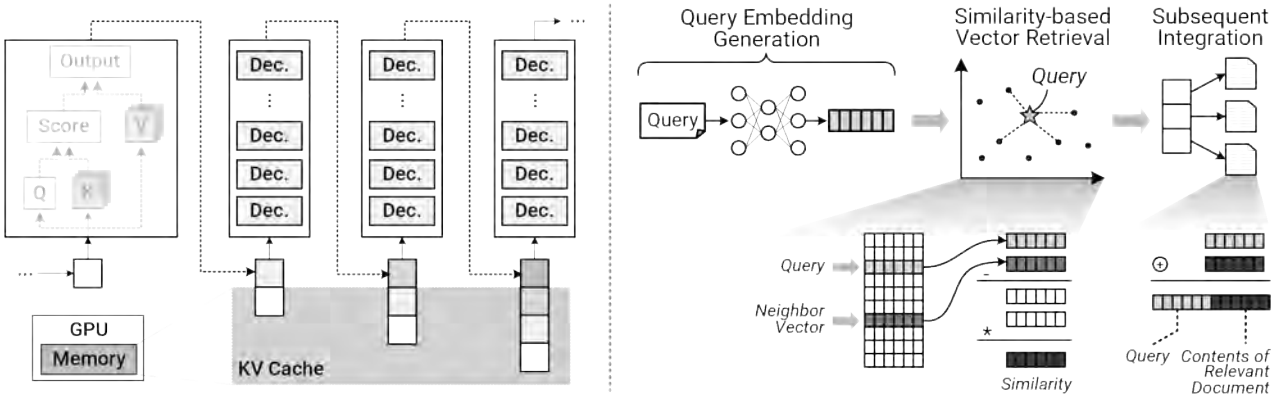
마찬가지로, 추론 단계에서도 앞서 언급된 오토레그레시브 방식은 동일한 원리로 동작한다. 그림 9b에 나타난 것처럼 오토레그레시브 추론 방식은 각 단계에서 생성된 토큰을 즉시 다음 예측에 필요한 입력 문맥으로 활용한다. 출력을 입력으로 사용하는 이러한 구조는 결과의 문맥적 정확성과 일관성을 높이지만, 병렬 연산의 기회를 제한하여 병렬 방식에 비해 상대적으로 추론 속도를 느리게 만든다. 이러한 연산 성능상의 제약에도 불구하고, 오토레그레시브 방식은 복잡한 언어적 종속성을 정확히 모델링하고 문맥에 맞는 고품질의 출력을 생성할 수 있기에 다양한 언어 생성 작업에서 널리 활용되고 있다.

반면, 오토레그레시브등이 가져다주는 연산적 한계를 보완하기 위한 한 방편으로, 현대의 LLM은 광범위한 사전 학습을 통해 높은 수준의 일반화 능력(Generalization Capability)을 확보하는 방식을 적용하기도 한다. 예를 들어 제로샷(Zero-Shot) 또는 퓨샷(Few-Shot) 학습 방식 [98, 175, 176]을 통해서 방대하고 다양한 텍스트 데이터를 통해 풍부한 언어적 표현을 내재화한 모델은 별도의 추가 학습이 없거나 최소한의 추가 학습만으로도 여러 후속 태스크(Downstream Task)에 유연하게 대응할 수 있도록 한다. 이렇게 확보된 일반화 능력은 전통적인 자연어 처리 분야를 넘어 이미지 생성, 비디오 합성, 오디오 처리 및 대화형 시스템과 같은 다양한 다중모달 응용 분야로 LLM의 활용 범위를 확장하는 데 중요한 기반을 제공하고 있다.

LLM 추론 과정의 중복 연산 최소화 및 신뢰성 향상. LLM이 다양한 응용 분야에 폭넓게 활용되면서, 추론 과정에서의 계산 효율성 및 정확도 문제를 해결하기 위한 두 가지 핵심 기술이 제안되었다. 바로 "KV 캐싱(Key-Value Caching [66, 67, 88, 177])"과 "검색 기반 생성(RAG, Retrieval-Augmented Generation [41, 42, 63])"이다.

KV 캐싱은 긴 시퀀스 처리 시 이전 결과를 메모리에 저장하여 반복 계산을 피하고 추론 성능을 높일 수 있지만, GPU 메모리 사용량이 크게 증가한다는 단점이 있다.

먼저, KV 캐싱은 LLM의 셀프어텐션 연산 과정에서 발생하는 중복 계산을 줄이기 위한 기술로 상당한 연산 속도의 개선을 가져다준다. 다시 말해, 오토레그레시브 추론에서는 각 토큰을 생성할 때마다 이전 토큰들과의 관계를 반복적으로 계산해야 하는데, 이는 입력 시퀀스 길이가 길어질수록 급격히 증가하고 별도의 최적화가 없으면 심각한 성능 저하로 이어질 수 밖에 없다. 이러한 문제를 해결하기 위해, 그림 10a와 같이, KV 캐싱은 처음 계산된 셀프어텐션의 결과를 키-값 형태로 GPU 메모리에 저장하고, 이후 추론 단계에서 이를 재사용함으로써 중복 연산을 방지한다. 따라서 KV 캐싱은 긴 시퀀스 처리 시 메모리에 저장된 이전에 만들어진 결과를 재사용하여 애초에 관련된 반복 계산 자체를 피하고 추론 성능을 현저히 개선할 수 있다. 그러나 이러한 데이터 저장을 통한 계산 대체 방식은 GPU 메모리 사용량을 상당히 증가시킨다는 단점이 발생할 수밖에 없다. 모델 크기, 입력 토큰 수, 그리고 추론의 복잡성에 따라 KV 캐싱 데이터는 GPU 메모리의 30%에서 최대 75%를 차지하며 [29–31, 178], 필수 데이터들이 단일 GPU 메모리에



(a) GPU 메모리에 캐싱되는 키-값 쌍. **(b)** 외부 데이터베이스 검색.

|그림 10| 추론 최적화 기법.

수용되지 않는 경우를 빈번하게 만든다.

반면에, KV 캐싱이 계산효율성을 극대화하기 위한 기법이라면, RAG는 LLM이 본질적으로 가지고 있는 "환각 현상"(Model Hallucination)을 방지하기 위한 기술이다 [179, 180]. 환각 현상은 모델이 실제 존재하지 않거나 잘못된 정보를 사실처럼 생성하는 문제로, 쉽게 이야기하면 LLM이 학습 과정에서 얻은 정적인 지식만으로 실시간 외부 정보를 반영하지 못하기 때문에 발생한다고 볼 수 있다. 이러한 한계를 극복하고자, RAG는 추론 과정에서 외부의 지식 데이터를 실시간으로 검색해 결과에 반영하는 방식을 사용한다. 그림 10b에처럼, 입력 쿼리가 주어지는 경우에 대해서 RAG를 적용한 LLM은 벡터 데이터베이스(Vector Database) [181–183]나 검색 시스템 [62, 184–186]에서 관련 최신 정보를 먼저 찾아낸다. 검색된 정보는 입력과 결합되어 최종 답변 생성에 활용되며, 이를 통해 모델의 환각 현상을 줄이고 결과의 정확도를 높일 수 있다.

그러나 RAG 역시 쿼리 임베딩 생성, 벡터 기반 유사도 검색, 검색 결과 통합 등 여러 단계의 추가 연산을 필요로 하므로, 아쉽게도 어쩔 수 없이 전체 시스템의 복잡도가 증가할 수밖에 없다. 또한, 대규모 벡터 데이터베이스 운영을 위한 추가 메모리 자원이 요구되며, 외부 정보 검색 성능은 응답의 지연과 정확도에 직접적인 영향을 미치기 때문에 네트워크 지연과 대역폭도 중요한 성능 지표로 고려해야 한다.

RAG는 외부 지식을 실시간으로 반영해 LLM의 환각 현상을 방지하지만, 추가 연산과 메모리 자원을 요구하며 네트워크 지연과 대역폭도 중요하게 고려해야 한다.

분명, KV 캐싱은 셀프어텐션의 중복 연산을 최소화해 추론 속도를 높이고, RAG는 외부의 실시간 정보를 결합하여 생성 결과의 신뢰성과 정확성을 높이는 것으로, 이 둘은 모두 LLM 추론 과정의 주요 병목 현상을 효과적으로 해결할 수 있는 기술이다. 하지만, KV 캐싱과 RAG 두 기술은 GPU 메모리와 계산 자원, 네트워크 대역폭 및 저장 장치 인프

라에 상당한 부담을 초래함도 사실이다. 따라서 LLM 워크로드를 효과적으로 지원하기 위해서는 고대역폭, 저지연 인터커넥트 기술 등을 잘 적용하여 가속기 간 망을 구성해야 하며 AI 인프라 내의 가속기와 메모리들이 높은 확장성을 가질 수 있도록 컴포저블 구조를 갖춘 데이터센터 아키텍처가 필수적으로 요구되는 방향으로 전환이 일어나고 있다.

3. LLM 확장: 다중 가속기 시스템에서 데이터센터 규모 인프라까지

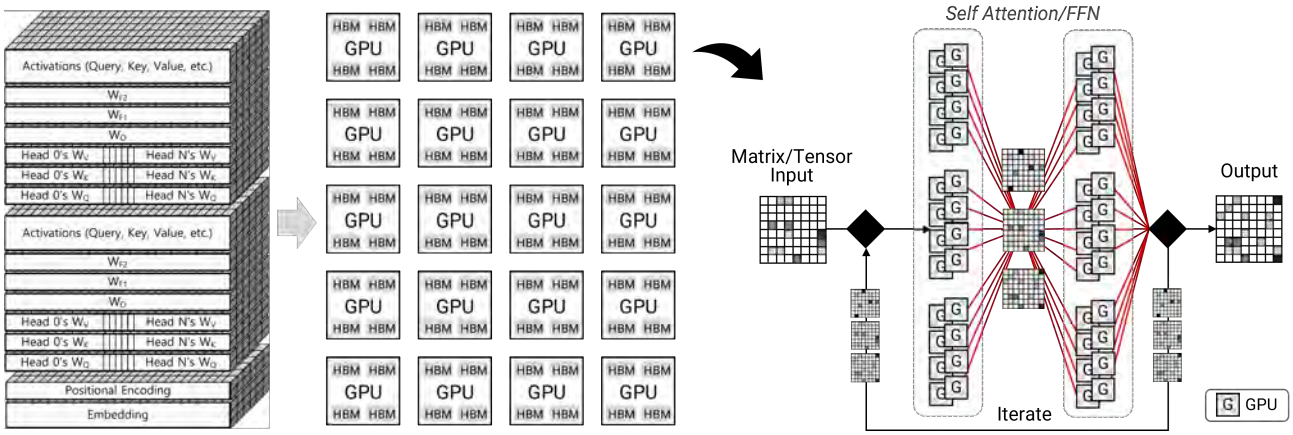
현대 데이터센터는 추천 시스템 [187–189], 랭킹 알고리즘 [190, 191], 비전 모델 [192, 193] 등 다양한 AI 워크로드를 처리할 수 있도록 지속적으로 발전해 왔다. 그러나 LLM의 경우, 좀 더 높은 메모리 및 통신 수요를 발생하고 이로 인해 기존의 다른 워크로드와 비교했을 때 데이터센터 인프라에 훨씬 더 큰 부담을 준다. 이를 좀 더 심도 있게 이해하기 위해, 본 절에서는 먼저 기본적인 LLM 개념이 다중 가속기 시스템(Multi-Accelerator Systems) 환경에서 어떻게 구현되고 있는지 살펴볼 것이다. 이어서, 현대의 데이터센터가 수천 개의 GPU 또는 가속기[4]를 활용하기 위해 어떤 아키텍처적 및 모듈화 전략을 채택하고 있는지 분석한다.

마지막으로, CPU와 GPU가 밀접하게 통합된(Tightly-Integrated) 아키텍처가 가지는 근본적인 한계를 논의한다. 이러한 CPU-GPU 밀접 통합 구조는 데이터센터의 확장성, 유연성, 효율적인 자원 활용에 제약이 생기는데, 이 제약을 극복하고 대규모 AI 워크로드의 요구사항을 충족시키기 위해서는 CPU, GPU, 메모리 및 네트워크 구성 요소가 독립적으로 확장 가능하도록 모듈화된 설계가 필요하다는 점을 논의할 것이다.

3.1. 다중 가속기 시스템에서의 LLM 적용과 도전 과제

현대 LLM은 그 규모가 지수적으로 커지면서, 단일 GPU가 제공 가능한 메모리 용량과 연산 성능의 한계를 이미 넘어섰다 [43, 44, 194]. 이에 따라, 데이터센터에서는 대규모 모델을 수천에서 수십만의 GPU에 분산 배치하여 병렬 실행하는 구조가 일반적이 되어가고 있다. 이러한 다중 GPU 환경에서 각 GPU는 전체 모델 중 특정 부분의

[4]본 기술 보고서의 기술 개요에서 설명 되었듯이, 본 절 분석에서는 GPU와 가속기라는 용어는 혼용하여 사용한다.



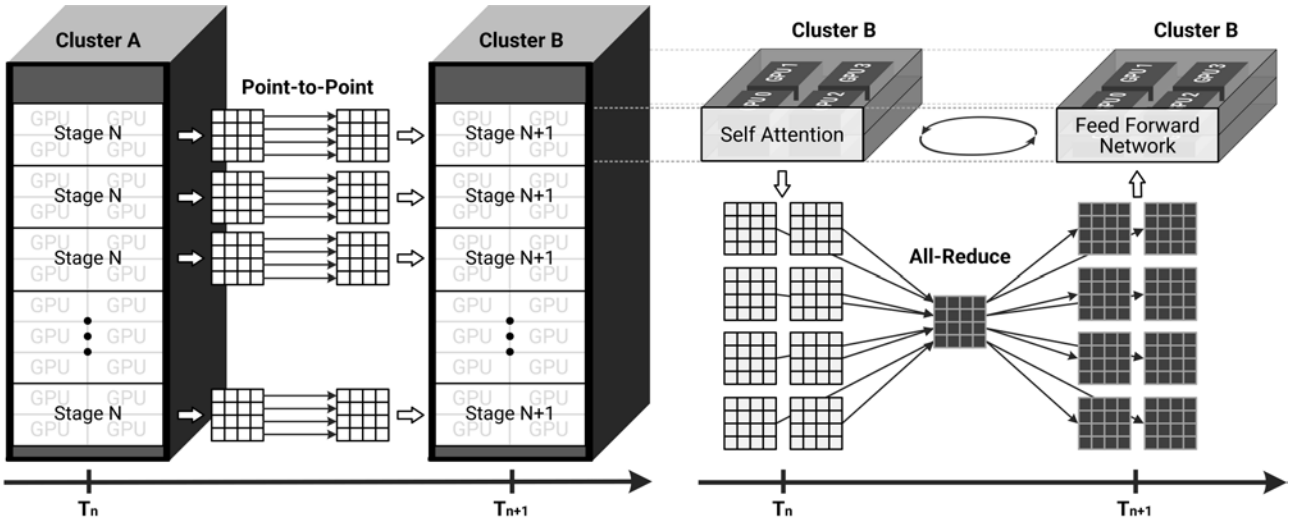
|그림 11| 트랜스포머 기반 모델에서의 텐서 분할과 동기화.

파라미터와 연산만을 담당하되 수많은 GPU가 동시에 하나의 문제를 풀 수 있도록 하는 형태로 실행된다. 이러한 실행을 위해 효과적인 병렬 처리와 분산 학습이 가능하도록 다양한 방법들이 사용되고 있다.

다중 GPU 환경에서의 LLM 학습: 모델 분할, 병렬화, 그리고 오버헤드. 다중 GPU 기반의 LLM 학습에서 가장 중요한 과제는 거대한 모델 파라미터, 활성화값(Activation), 그리고 그래디언트를 수많은 GPU들에 효율적으로 분할하고 동기화하는 것이라 해도 무방할 정도이다. 다수의 GPU에 분할, 병렬 실행하는 동안에도 연산의 일관성을 유지하면서 정확하고 효과적인 분산 학습을 가능하게 해야 한다. 그림 11은 트랜스포머 기반 LLM 모델에서의 GPU 간 "텐서(Tensor)" 분할 및 동기화 전략을 나타내는데, 특히 셀프어텐션과 FFN 계층에서 발생하는 중간 결과가 GPU 간에 빈번하게 교환되는 과정을 좀 더 자세히 나타내었다. 앞서 거듭 설명되었지만, 트랜스포머의 셀프어텐션 메커니즘은 시퀀스 내 모든 토큰 간 상호작용을 계산하게 되어 있다. 이는 언뜻 보기에는 쿼리, 키, 밸류 벡터를 GPU 간에 나눠서 계산하고 전달해야 할 것 같지만, 실제로는 멀티 헤드 어텐션과 쿼리 그룹 어텐션과 같은 구조를 통해 전체 어텐션 계층을 여러 개의 독립적인 소규모 계층으로 병렬화할 수 있도록 설계되어 있다고 볼 수 있다. 따라서 각 GPU는 자신에게 할당된 소규모 어텐션 계산을 독립적으로 수행하기 때문에 다중 가속기 시스템에 매우 최적화되어 있다. 이러한 병렬 실행 구조에도 불구하고 실제 어텐션 처리는 LLM의 최대 병목 중 하나인데 그 이유는 각 GPU가 그 계산 결과와 그래디언트를 다른 GPU들과 주기적이면서도 빈번히 동기화해야 하기 때문이다 [29, 195, 196]. 이러한 동기화 과정은 모델 전체의 일관성과 수렴성을 보장하기 위한 필수 조건이기 때문에 생략될 수 없으며, 이로 인해 다중 가속기 시스템에서 GPU 간 통신 대역폭과 메모리 자원에 큰 부담을 주게 된다.

다중 GPU 기반 LLM 학습의 주요 과제는 거대한 모델 파라미터, 활성화값, 그래디언트를 다수의 GPU에 효율적으로 분할하고 동기화하는 것이다.

셀프어텐션 외에도, 상위 절에서 논의되었던 것처럼 트랜스포머는 토큰 단위로 독립적인 연산을 수행하는 FFN



(a) 파이프라인 병렬화. **(b)** 텐서 병렬화.

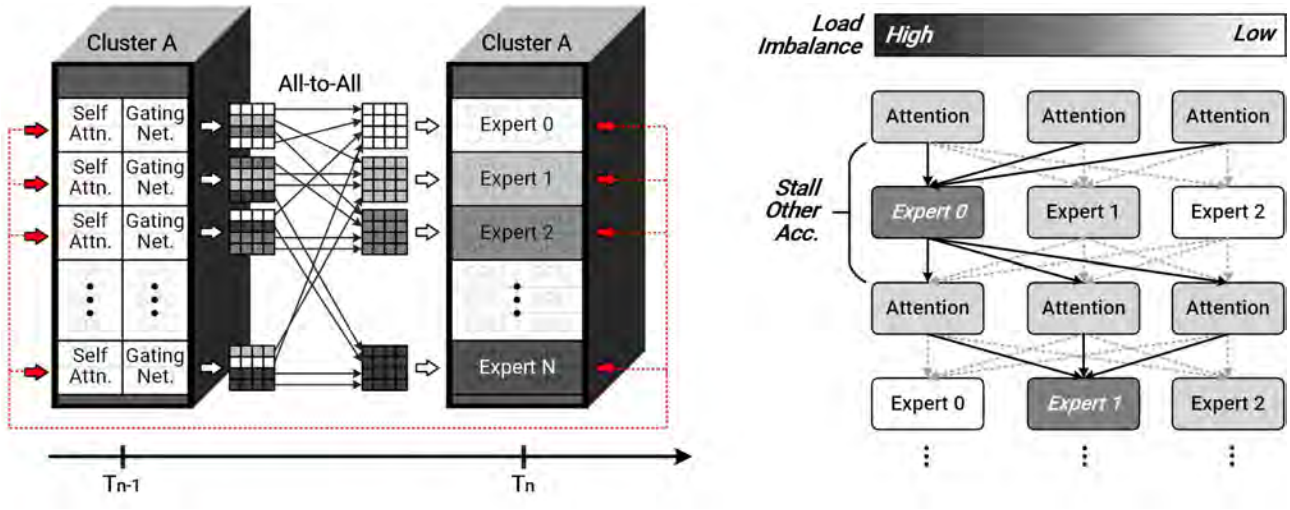
|그림 12| 트랜스포머 구조를 위한 파이프라인 병렬화와 텐서 병렬화 기법.

계층을 포함하고 있다. FFN의 연산 또한 병렬 처리에 적합하도록 설계되어 있지만, 아쉽게도 순전파(Forward Pass)와 역전파(Backward Pass) 단계에서는 중간 결과와 그래디언트를 GPU 간에 교환하여 동기화할 필요가 있다 [197–201]. 특히, 그래디언트 동기화는 매우 빈번히 일어나므로 다중 가속기 시스템에서 GPU 간 통신 오버헤드를 크게 증가시키는 것으로 잘 알려져 있다 [202–204].

이러한 GPU 간 통신 문제를 해결하기 위해, LLM 학습에서는 이미 파이프라인 병렬화(Pipeline Parallelism) 및 텐서 병렬화(Tensor Parallelism)와 같은 고급 병렬화 기법들이 중요하게 활용되고 있다 [43, 45, 170, 205, 206]. 그림 12a에 도식화된 것처럼, 파이프라인 병렬화는 모델을 여러 개의 연산 단계(Stage)로 나누고, 각 단계를 각기 다른 GPU 클러스터에서 순차적으로 처리하는 방식이다. 이를 통해 GPU나 가속기 같은 연산 자원들의 활용도를 극대화할 수 있지만, 각 연산 단계 간에 데이터 의존성이 존재하므로 정확한 데이터 교환과 동기화가 이루어지지 않으면 파이프라인 버블(Pipeline Bubble)이라는 유휴 시간이 발생하고 심각한 성능 저하를 야기할 수 있다 [45, 170, 207–209]. 따라서 GPU의 활용도를 높이기 위해서는 이러한 데이터 교환 타이밍을 세밀하게 조정하는 것이 중요하다. 일반적으로 파이프라인 병렬화는 각 단계의 연산량이 매우 크고, 단일 GPU의 연산 능력조차 초과하는 대규모 모델에 효과적으로 알려져 있다 [28, 43, 170].

한편, 텐서 병렬화는 파이프라인 병렬화의 보완책으로서, 행렬 곱(Matrix Multiplication)과 같이 규모가 큰 텐서 연산을 여러 GPU에서 동시에 수행하는 방식이다. 그림 12b는 이러한 텐서 병렬화 구조를 나타낸다. 텐서 병렬화는 올-리듀스(All-Reduce), 올-게더(All-Gather), 리듀스-스캐터(Reduce-Scatter) 등과 같은 집합 통신 연산(Collective Communication Operations)을 통해 GPU 간에 중간 연산 결과를 주기적으로 동기화하는데 이때 GPU 간 이동해야 하는 데이터가 매우 방대하다 [32, 210–215]. 집합 통신 연산에서 GPU 간 동기화는 분산된 연산 결과의 일관성을 유지하기 위한 필수 과정이며, 이에 대한 방대한 데이터 이동은 높은 성능의 인터커넥트 기술 및 최적화된 통신 알고리즘 지원 없이는 다중 가속기 시스템의 성능을 급격히 저하할 수 있다.

한편, MoE와 같은 동적으로 계산 자원을 할당하는 모델 구조가 등장하고 그 이용이 대중화됨으로써, 학습 워크



(a) 전문가 병렬화의 실행 방식.

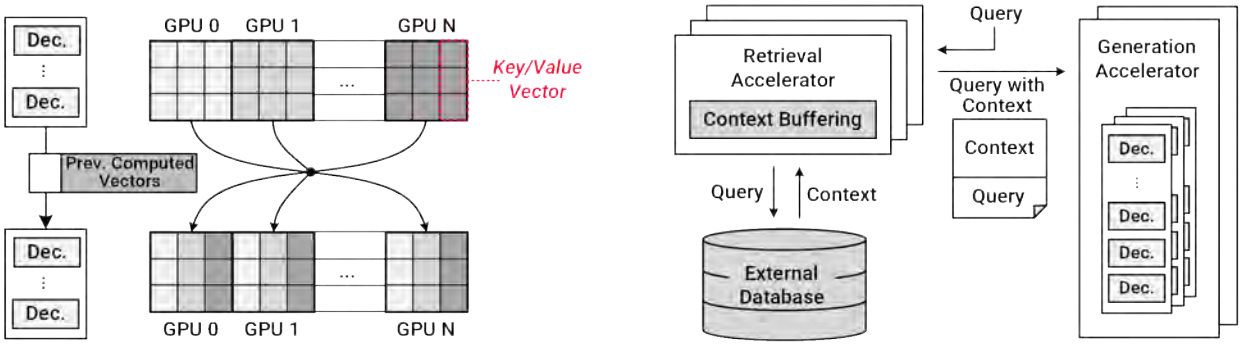
(b) 전문가 병렬화의 동기화 오버헤드.

|그림 13| MoE 구조를 위한 전문가 병렬화 기법.

플로우의 복잡성이 현저히 높아지는 것도 중요한 문제 중 하나이다 [162, 216]. 그림 13a는 다중 가속기 시스템이 MoE 전문가 모듈을 GPU 간에 분산하여 구성하고 실행하는 방법을 보여준다. 그림에 나타나 있듯, MoE 구조에서 각 GPU는 개별적인 전문가 역할을 수행하며, 특정 입력 데이터의 하위 집합에 대해 독립적인 순전파 및 역전파 연산을 처리한다 [158, 162, 217–219]. 입력 시퀀스는 미리 정의된 기준 또는 라우팅(Routing) 전략에 따라 적절한 GPU로 분배되는 방식으로 스케줄링(Scheduling)된다. 예를 들어, 문장이나 질의 내의 특정 부분(토큰 또는 세그먼트)은 모델의 전문가 선택 정책에 따라 미리 지정된 GPU로 전달되며, 각 GPU는 독립적으로 할당된 연산을 수행하여 다중 가속 시스템의 연산 병렬성을 최대화한다.

하지만 최종적으로 의미 있는 예측 결과를 얻기 위해서는 각 전문가가 처리한 중간 결과를 GPU 간에 반드시 통합하고 동기화해야 하므로 수많은 GPU가 멈춰야 하는 오버헤드를 피할 수 없다는 문제가 있다. 그림 13b에서 알 수 있듯이, MoE 구조에서 중간 결과의 집계 및 동기화 과정은 수행 중인 모든 전문가와 GPU가 데이터 이동이 끝나길 기다리기 때문에 매우 비싼 작업이며, 이러한 동기화 작업은 전체 모델의 일관성을 유지하기 위해 전문가 간 그래디언트 통합을 포함하여 아주 빈번히 발생한다. 따라서 이러한 MoE 모델의 학습은 필연적으로 다중 시스템에서 GPU 간 통신 부담을 극단적으로 증가시키기 때문에, 이러한 학습 등을 수용하기 위해서라도 새로운 AI 인프라는 높은 대역폭과 낮은 지연을 보장하는 정교한 인터커넥트 기술의 설계와 도입이 필수적이다.

MoE와 같은 병렬 모델은 GPU 간 통신 부담을 크게 증가시키므로, 높은 대역폭과 낮은 지연을 제공하는 정교한 인터커넥트 설계가 필수적이다.



(a) GPU 간 키, 밸류 벡터 재사용을 위한 KV 캐싱. **(b)** 대규모 검색 기반의 RAG.

|그림 14| 추론 최적화 기법의 통신 오버헤드와 메모리 오버헤드.

다중 GPU 환경에서의 LLM 추론: 최적화 기술과 도전 과제. LLM 학습과 달리 LLM의 추론 단계는 연산 속도와 실시간 응답성의 성능을 중요시하는 워크로드들로 구성되어 있는 경우가 많다 [220–222]. 과거 이러한 상황에서는 고속 GPU나 도메인에 특화된 하드웨어 가속기들을 다수 수용함으로써 많이 해결할 수 있었다. 그러나 대규모의 입력과 모델 등의 영향에 따라 단순히 GPU의 계산 성능만으로 LLM의 복잡한 요구를 만족시키기 어려워짐에 따라 최근 다양한 추론 최적화 기법들 [223–225]이 제시되고 있다. 이러한 추론 최적화 기법의 도입과 모델 크기 증가에 따라, 오늘날 최적화된 실제 추론 경로에서는 GPU 간 "통신 대역폭"과 "메모리 용량"이 주요 성능 병목 요인으로 작용하고 있다.

최적화 기법의 도입과 모델 크기 증가에 따라, 오늘날 추론 경로에서도 GPU 간 통신 대역폭과 메모리 용량이 주요 성능 병목 요인으로 작용하고 있다.

이러한 병목 요인의 이해를 돕기 위해 그림 14는 가장 많이 사용되는 추론 최적화 기법인 KV 캐싱과 RAG를 적용했을 때 나타나는 다중 가속기 시스템의 GPU 간 통신 패턴 및 외부 데이터 접근 방식의 차이를 보여준다. 먼저, 그림 14a에 나타난 KV 캐싱은 앞서 언급된 것처럼 이미 계산된 쿼리, 키, 밸류 벡터를 GPU 메모리에 미리 저장하고 반복적인 추론 단계에서 저장된 값들을 재사용하여 중복 계산을 방지하는 기법이다. 오토레그레시브 방식을 포함 대부분의 LLM 추론에서는 각 토큰을 생성할 때마다 이전 모든 토큰과의 관계를 반복적으로 계산해야 하므로 불필요한 연산이 누적되는데, KV 캐싱은 이러한 중복 연산을 방지하여 성능을 상당히 개선할 수 있다 [33, 68]. 특히 긴 입력 시퀀스를 다룰 때 추론 지연을 매우 효과적으로 감소시킬 수 있다. 하지만 KV 캐싱은 전체 GPU 메모리의 30%에서 75%까지 차지하고 있으며, 모델의 크기와 컨텍스트 윈도우(Context Window)의 길이가 증가할수록 저장해야 하는 KV 캐시의 총량이 기하급수적으로 늘어나기 때문에, AI 인프라를 구축하는 데 있어서 상당한 문제로 주목받을 수밖에 없다. 현재 구조에서는 KV 캐싱이 단일 GPU 메모리가 수용할 수 있는 한계를 넘어 데이터를 관리해야 하므로,

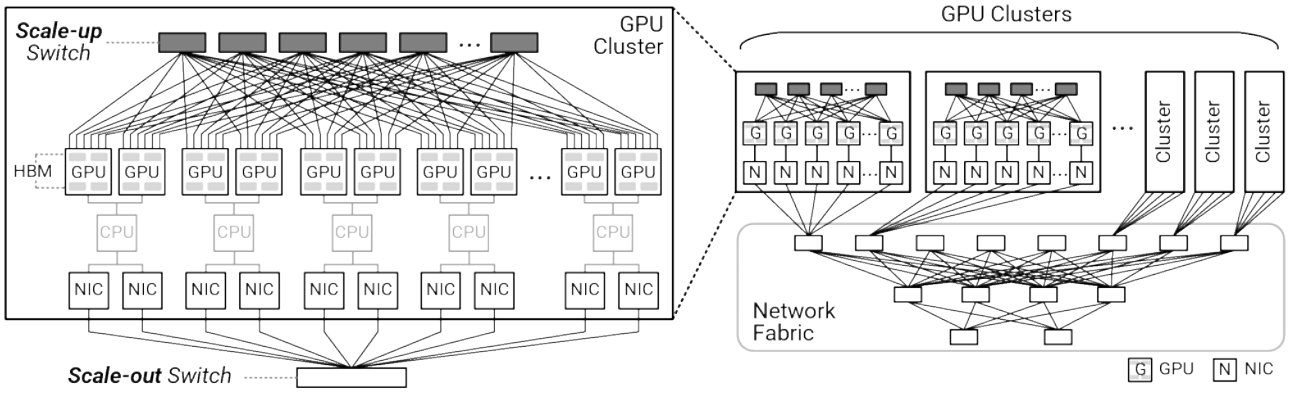
이러한 키-값 데이터를 여러 GPU에 분산 저장하고 일관성을 유지하기 위한 GPU 간 동기화를 빈번히 요구하게 되며, 이에 따라 전체 시스템의 메모리 관리 복잡도와 통신 오버헤드를 동시에 증가시키고 있다 [67, 226].

한편, RAG는 LLM의 정확성과 사실 기반 응답 신뢰도를 높이기 위해 도입된 기술로, KV 캐싱과는 또 다른 형태의 자원 부담과 복잡성을 초래한다. 그림 14b에서 보듯, RAG는 추론 단계에서 외부 지식베이스(예: 벡터 데이터베이스, 문서 저장소)로부터 관련 정보를 실시간으로 검색한 후, 이를 기반으로 LLM이 최종 응답을 생성하는 방식이다. 이 접근법은 앞서 언급된 것과 같이, 모델 자체가 학습한 내재적 지식만으로는 최신 정보나 문맥에 맞는 정확한 응답을 제공하기 어려워 발생하게 되는 환각 문제를 효과적으로 완화할 수 있다 [62, 227–229]. 구체적으로 RAG는 다음과 같은 세 단계로 구성된다: (1) 쿼리 임베딩 생성, (2) 벡터 유사도 기반 검색(Similarity-Based Retrieval), (3) 검색된 외부 정보와 원 입력을 결합하여 디코딩 수행. 이 과정들은 각각 별도의 연산 흐름 관리와 외부 메모리 접근을 요구하며, 이외에도 검색된 데이터를 일시적으로 저장하기 위한 추가적인 메모리 공간이 필요하다. 또한 외부 데이터 참조를 위해 GPU가 고속으로 외부 데이터베이스를 쿼리해야 하므로 비 RAG기반 시스템과 달리 네트워크 대역폭과 지연이 전체 추론 시간에 직접적인 영향을 준다. 이러한 네트워크 의존성은 시스템 구조와 아키텍처 설계에 많은 영향을 줄 수밖에 없다. 예를 들어, 벡터 검색을 수행하는 서버와 추론을 담당하는 GPU 간의 통신 지연이 길어질 경우, 모델은 검색 결과를 기다리며 유휴 상태에 머물게 되어 매우 비효율적인 운영을 하게 된다. 따라서 RAG 기반 시스템을 지원하는 AI 인프라 구조에서는 고성능 네트워크 인터페이스 카드(NIC, Network Interface Card)와 저지연 네트워크 패브릭(Network Fabric)을 필수적으로 사용하고 있으나 [41, 62, 230] 이러한 장거리 네트워크 요소들은 전체 시스템의 심각한 성능 저하를 유발하기에 최신 AI 인프라에서는 다수의 장치들을 연결할 수 있는 단거리 인터커넥트 기술들이 제시되고 다양한 시스템에 적용되고 있다.

추론에서는 KV 캐싱이나 RAG 같은 최적화 기술이 적용되어도 오토레그레시브 방식의 구조적 특성등으로 인해 제한되는 성능을 개선할 수 있는 것에 한계가 존재한다. 오토레그레시브는 앞서 논의된 것처럼 시퀀스 생성 시 각 토큰이 이전에 생성된 모든 토큰에 명시적으로 의존하게 되므로, 각 단계가 반드시 순차적으로 수행되어야 한다는 단점이 있다. 이는 토큰 생성 과정을 병렬화하기 매우 어렵게 만들며, 이를 해결하기 위해 다중 가속기 시스템은 필수적으로 오토레그레시브 작업 등의 중간 계산 결과를 관련 GPU들끼리 신속히 교환하게 하고 정밀히 동기화시켜야 한다. 따라서 연산 속도와 응답성과 같은 가속기 성능 지표에만 의존하던 과거 추론 달리 현대 추론 시스템은 네트워크 통신 속도뿐 아니라 GPU 간의 낮은 연결 지연을 보장하도록 가속기 간 인터커넥트 성능과 효율성 등에 신경 써야 한다.

KV 캐싱은 중복 연산을 줄이지만 GPU 메모리 소모를 늘리고, RAG는 신뢰성을 높이지만 통신과 저장소 부담을 키운다. 오토레그레시브 구조는 문맥 일관성을 제공하나 병렬성 제한과 동기화 오버헤드가 따른다.

종합적으로 생각해 보면, 현대의 LLM 추론 환경은 단순한 계산 성능만으로 충족할 수 없는 다양한 시스템 수준의 요구사항을 동반할 수밖에 없다. KV 캐싱은 중복 연산을 줄여주지만, GPU 메모리 소모를 증가시키며, RAG는 응답의



(a) 스케일업과 스케일아웃 아키텍처.

(b) 네트워크 패브릭으로 연결된 GPU 클러스터.

|그림 15| 다중 가속기 시스템의 스케일업과 스케일아웃 전략 및 그 구조.

신뢰성을 높이는 대신 외부 데이터 접근 및 통합 처리에 따른 통신과 저장소 요구를 가중한다. 오토레그레시브 구조는 문맥적 일관성을 제공하지만, 병렬 처리의 제한과 동기화 오버헤드를 수반한다는 문제를 동시에 가진다.

따라서 현대적인 LLM 추론 워크로드를 효과적으로 처리하려면 단순히 빠른 GPU나 특정 응용이나 도메인에 특화된 가속기를 추가하는 접근이 아니라, 전체 아키텍처 수준에서 메모리 계층 구조(Memory Hierarchy), 네트워크 인터커넥트, 캐시 일관성 유지 전략 등을 포괄적으로 재설계해야 한다. 특히 지연, 처리량, 메모리 활용률, 통신 효율성과 같은 다양한 성능 지표를 균형 있게 고려하는 모듈형 및 컴포저블 인프라 설계가 매우 중요해지는 시점이다.

3.2. 다중 가속기 시스템의 확장: 스케일업과 스케일아웃 아키텍처

다수 GPU를 사용하는 학습과 추론 과정에서 발생하는 GPU 간 통신 문제를 해결하려면, 성능 및 확장성 요구 사항에 맞추어 설계된 정교한 하드웨어 인터커넥트와 네트워크 솔루션이 필수적이다. 이때 사용되는 구조적 구성 전략은 주로 "스케일업(Scale-up)"과 "스케일아웃(Scale-out)"으로 나누어진다. 데이터센터의 스케일업 아키텍처는 NVLink [71–73], NVLink Fusion [74, 75, 231, 232], UALink [69, 70], CXL [54–56]과 같은 고속의 특화된 인터커넥트를 사용하는 반면, 스케일아웃 아키텍처는 Ethernet [233–236]이나 InfiniBand [237–240]와 같은 장거리 고대역폭 네트워크를 활용한다.

스케일업(Scale-up) 아키텍처: 고속 직접 연결 방식. 그림 15a는 다중 가속기 시스템 환경에서 스케일업과 스케일아웃 GPU 인터커넥트 아키텍처를 비교하여 보여준다. 스케일업 방식은 소수의 GPU를 특수 설계된 고속 가속기 중심(Accelerator-Centric) 인터커넥트로 긴밀하게 연결한다. 이러한 직접적이고 높은 대역폭의 연결은 긴밀하게 결합된 GPU 클러스터 내에서 빈번하고 많은 양의 데이터 교환이 필요한 작업에 특히 효과적이다. 따라서 현재 스케일업 방식은 노드 내부의 GPU 간 통신 속도를 극대화하고 지연 시간을 최소화해야 하는 워크로드에 적합하며, 학습, 실시간 추론, GPU 연산 중심의 AI 작업 등 계산 집약적인 작업의 성능을 크게 향상시킨다.

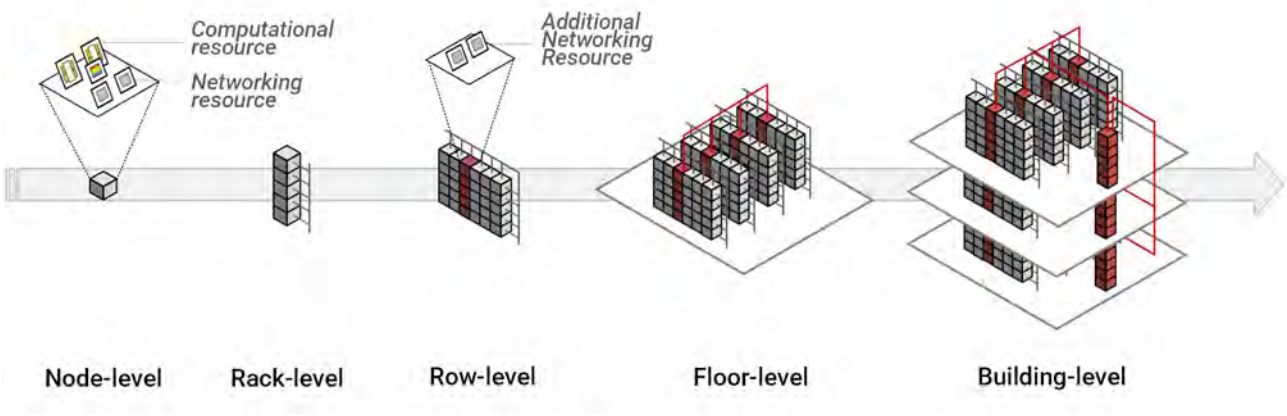
LLM과 최근의 데이터센터 아키텍처가 등장하기 이전에는, 이러한 긴밀한 직접 연결 방식을 필요로 하는 GPU의 수가 상대적으로 제한적이었다고 할 수 있다. 그러나 최근 들어 모델의 복잡도가 증가하고 데이터양도 급격히 늘어나면서 [173, 194], 고속 연결성을 요구하는 GPU의 수가 기하급수적으로 증가하고 있다. 더욱이 최신 LLM 최적화 기법들 중 상당수는 고속·저지연 데이터 교환과 지속적인 I/O 데이터 공유를 필요로 한다 [24, 28, 43, 45, 170, 241–243]. 그 결과, 현대 데이터센터에서는 더 많은 GPU를 랙(Rack) 단위로 밀집 배치하여 계산 효율성을 극대화하고, 통신 오버헤드를 최소화하며, 총소유비용(TCO, Total Cost of Ownership)을 낮추는 경향이 뚜렷해지고 있다. 예를 들어, NVIDIA는 고밀도 GPU 배치를 위해 콤팩트하고 수랭식으로 설계된 새로운 노드 유닛을 제안하고 고속 NVLink 및 NVSwitch 인터커넥트를 활용하여 랙당 최대 72개의 GPU를 긴밀하게 통합하여 데이터센터 업체들에게 제공을 시작하였다 [244]. 이와 같은 고속 직접 연결 방식의 배치는 계산 처리량을 높이고 GPU 간 통신 효율성을 최적화하는 동시에, 열 관리를 향상시켜 운영 복잡성을 줄이고 TCO를 감소시키는 장점이 있다.

스케일업은 고속의 직접 연결방식으로 다수의 가속기를 연결하는 반면, 스케일아웃은 장거리 네트워크 패브릭을 통해 연결하는 방식이다. 최근 LLM과 같이 데이터 교환과 동기화가 빈번한 워크로드가 많아지면서, 스케일업 방식의 중요성이 더욱 커지고 있다.

스케일아웃(Scale-out) 아키텍처: 장거리 네트워크 인터페이스. 반면 스케일아웃 구성 방식은 수천에서 수십만 개의 GPU가 여러 랙 또는 노드에 걸쳐 분산되는 데이터센터 수준의 대규모 환경을 위해 설계되었다. 광범위한 확장성과 유연한 자원 관리가 가능하도록 스케일아웃 아키텍처는 주로 NIC과 RDMA 기반의 통신 프로토콜을 활용하여 GPU 간의 상호작용을 지원하는 "장거리(Long-Distance)" 네트워크 패브릭을 사용한다 [245–247]. 그림 15b에서 나타낸 것처럼, 스케일아웃에서 GPU는 원격 네트워크 패브릭을 통해 상호 연결된 클러스터 형태로 구성되어 있으며, 전체 시스템은 이러한 다수 GPU들을 그룹화하고 동적인 시스템 구성이 가능하도록 설계된다.

장거리 네트워크 기반의 패브릭은 뛰어난 확장성과 유연성 및 높은 집합 대역폭(Aggregate Bandwidth)을 제공하지만, 심각한 오버헤드를 수반한다는 단점이 있다. 이 네트워크 패브릭 오버헤드는 복잡한 하드웨어 설계, 정교한 네트워크 프로토콜 및 소프트웨어 기반의 통신으로 인해 발생한다. 구체적으로, 데이터의 직렬화 및 역직렬화(Serialization/Deserialization), 네트워크 프로토콜 처리, 그리고 소프트웨어 수준의 상호작용으로 인해, 긴밀하게 결합한 하드웨어 기반의 스케일업 아키텍처에 비해 통신 지연이 현저하게 증가할 수 밖에 없다 [52, 53, 61]. 따라서 대규모 분산형 AI 시스템 설계 시에는 이러한 스케일업과 스케일아웃 사이의 성능과 효율성 간의 절충안(Trade-Off)을 면밀히 평가하고 고려하는 것이 필수적이다.

한편, 현재 AI 인프라에서 전통적인 CPU도 스케일업 및 스케일아웃 시스템 모두에서 대규모 다중 가속기 시스템을 지원하는 데 여전히 필수적인 역할을 수행한다. GPU가 주된 연산 작업을 담당하는 동안 CPU는 시스템의 오케스트레이션 역할을 수행하며, GPU 간의 조정, 데이터 전송 및 네트워크 인터페이스 관리를 담당한다 [210, 248–250]. 결과적으로 각 GPU 또는 GPU 클러스터는 하나 이상의 CPU와 NIC를 핵심 구성 요소로 통합되는 방식을 많이



|그림 16| 데이터센터의 계층적 구조.

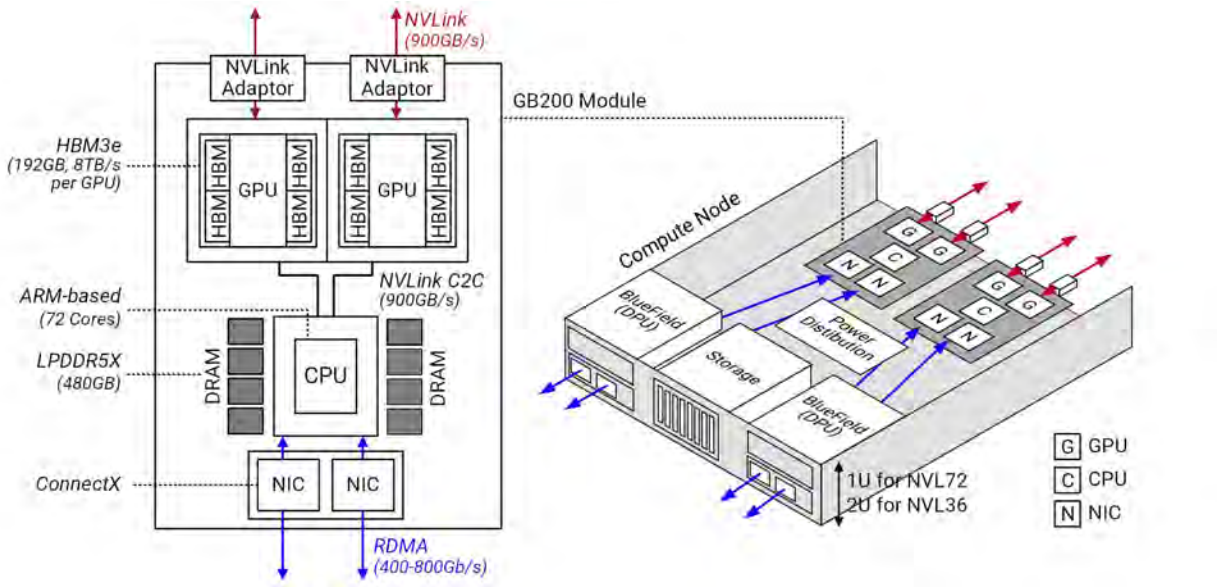
쓰고 있다. 스케일업과 스케일아웃 두 영역에서 모두, 과거에도 본 기술 보고서에서 제시된 방식과 유사하게 자원 분리(Disaggregation)를 구현하려는 시도는 존재했다. 그러나 CPU는 버스 인터페이스와 메모리 컨트롤러를 관리하면서 GPU 또는 가속기와 직접 연결되는 호스트 프로세서 역할을 수행해야 하므로, 완전한 물리적 자원 분리는 현실적으로 이루어 진적이 없다 [251–254]. 최근의 산업 트렌드는 오히려 앞서 이야기한 것처럼 단일 노드 내에서의 더욱 긴밀한 통합을 강조하거나, 모듈 단위로 확장하는 방식을 채택하고 있다. 이러한 노드 수준에서의 통합 사례로는 NVIDIA의 최신 GPU 모델(Grace Blackwell)이 대표적이며, 이에 대해서는 다음 절에서 추가로 살펴볼 예정이다.

3.3. 대규모 AI 인프라: 그레이스 블랙웰 아키텍처 기반의 계층형 데이터센터 구조

실제 대규모 데이터센터는 다양한 워크로드 특성과 이로 인해 변화하는 인프라 요구사항을 효과적으로 처리하기 위해 계층형(Hierarchical) 아키텍처를 많이 사용하고 있다. 그림 16은 데이터센터의 최소 구성 단위부터 이를 조합하여 하나의 큰 구조물 단위까지를 걸쳐 설계되는 계층적 구조를 단계별로 도식화 하고 있다. 이 구조는 컴퓨팅과 네트워크 자원을 여러 계층으로 나누어 추상화하며, 각 계층은 하위부터 상위까지 "노드(Node)", "랙(Rack)", "열(Row)", "층(Floor)", 그리고 최종적으로 "건물(Building)" 단위로 설계하여 다중 가속기 시스템을 대규모 형태로 구성할 수 있게 한다 [255–259].

그림에서 볼 수 있듯이, 데이터센터의 가장 하위 계층인 노드는 컴퓨팅의 기본 단위로, CPU, GPU, 메모리, 네트워크 인터페이스 등 필수적인 연산 요소들을 통합한 형태이다. 이 노드들이 집적된 랙은 연산 요소들의 계산 밀도를 높이고 네트워크 연결의 효율성을 향상시키기 위해 다양한 방법으로 구성될 수 있다. 다만 하나의 랙에 수용할 수 있는 노드 수는 랙의 물리적 크기에 한정되므로 [260, 261], 데이터센터는 다수의 랙을 모아 열[5]을 형성시키며, 이러한 다수의 열을 다시 층 단위로 결합시키는 방식으로 수많은 GPU들을 조직화한다. 최종적으로 여러 층의 조합이 단일 건물

[5]특정 벤더의 랙스케일 설계에서는 이를 슈퍼파드(Superpod)라고 부르기도 한다.



(a) 블랙웰 아키텍처 (GB200 모듈). **(b)** 두 개의 GB200 모듈이 탑재된 컴퓨트 노드.

|그림 17| GB200의 노드 수준 구성.

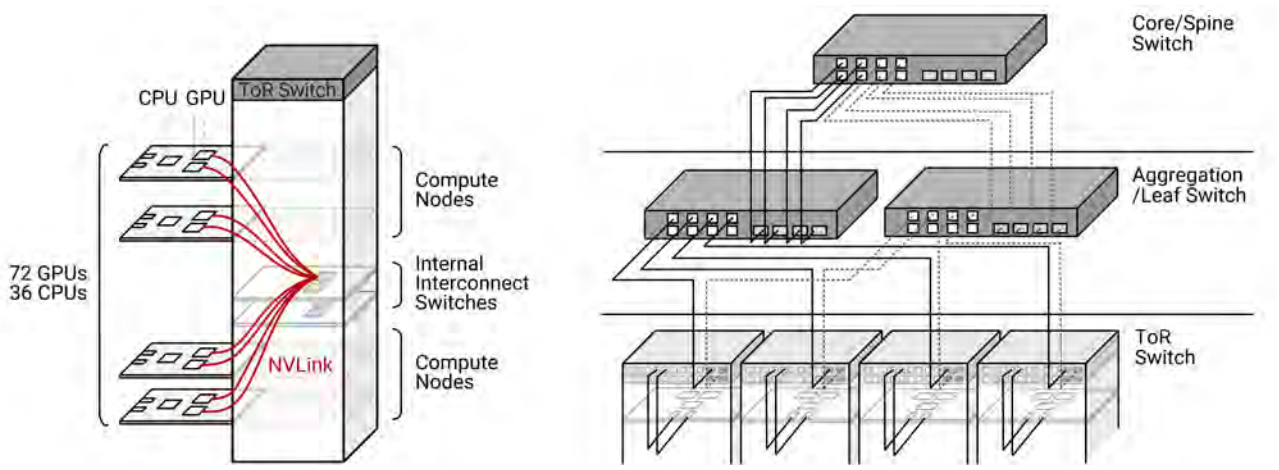
수준의 통합 인프라를 완성하여 고밀도 및 대규모 AI 클러스터 구축을 가능하게 한다.

본 절에서는 노드 수준의 구성에서 출발하여 랙, 열, 층, 그리고 마지막 건물 단위까지 확장하면서, 각 단계에서 컴퓨팅 및 네트워크 자원이 어떻게 통합되고 최적화되는지를 순차적으로 설명한다. 특히 본 절에서는 NVIDIA의 최신 그레이스 블랙웰 아키텍처 [20, 21, 244, 262, 263]를 노드 구성의 대표적인 예시로 활용하여, 현대 데이터센터 설계의 핵심 개념을 구체적으로 설명하고자 한다. NVIDIA의 블랙웰 아키텍처는 GPU당 HBM [36–38, 264] 용량을 확장하고 CPU-GPU 간 통합 구조를 강화하여 메모리 관리, GPU 간 통신, 연산 조정과 관련된 병목 해소를 목표로 설계된 대표적인 데이터센터 아키텍처로, 최신 대규모 AI 인프라에 널리 활용되고 있다.

현 기술 보고서에서 블랙웰 GPU의 구체적인 사례를 통해 개념 이해를 돕고는 있지만, 본 절의 목적은 특정 하드웨어 사례에 국한하지 않고 계층형 데이터센터 아키텍처의 전반적 이해를 제공하는 데 있다. 따라서 블랙웰 아키텍처는 현대적 AI 인프라 설계의 예시로 사용되며, 이후 섹션에서는 보다 일반화된 구조적 통찰을 바탕으로 다양한 설계 접근 방식을 소개하고, 이에 따른 확장 가능성 및 제약 조건을 논의한다.

노드 및 랙 수준 구성: CPU-GPU 모듈의 계층적 통합. 앞서 소개된 것처럼, 현대 AI 데이터센터의 기본 구성 단위는 컴퓨트 노드로, 이는 CPU, GPU, 메모리, 네트워크 인터페이스가 통합된 독립적 연산 장치이다. 그림 17a는 이러한 노드를 구성할때 하드웨어로써 포함될 수 있는 NVIDIA의 그레이스 블랙웰, GB200 모듈 구조를 예시로 나타낸다. 그림을 보면 알 수 있듯이 하나의 GB200 모듈은 72개의 코어를 포함한 ARM 기반 CPU 1개와 GPU 2개로 구성되며, 모듈 내의 CPU와 GPU는 NVLink 칩투칩(C2C, Chip-to-Chip) 인터페이스를 통해 고속으로 연결된다 [231, 232].

각 GPU는 약 192GB 용량의 고대역폭 메모리(HBM3e)를 내장하고 있으며, GPU당 최대 8TB/s의 메모리 대역폭을 제공하여 대규모 AI 모델 학습 및 추론 워크로드에 적합하도록 구성되어 있다 [20, 21]. CPU는 최대 480GB



(a) NVSwitch를 통한 내부 연결. **(b)** ToR 스위치를 통한 외부 연결.

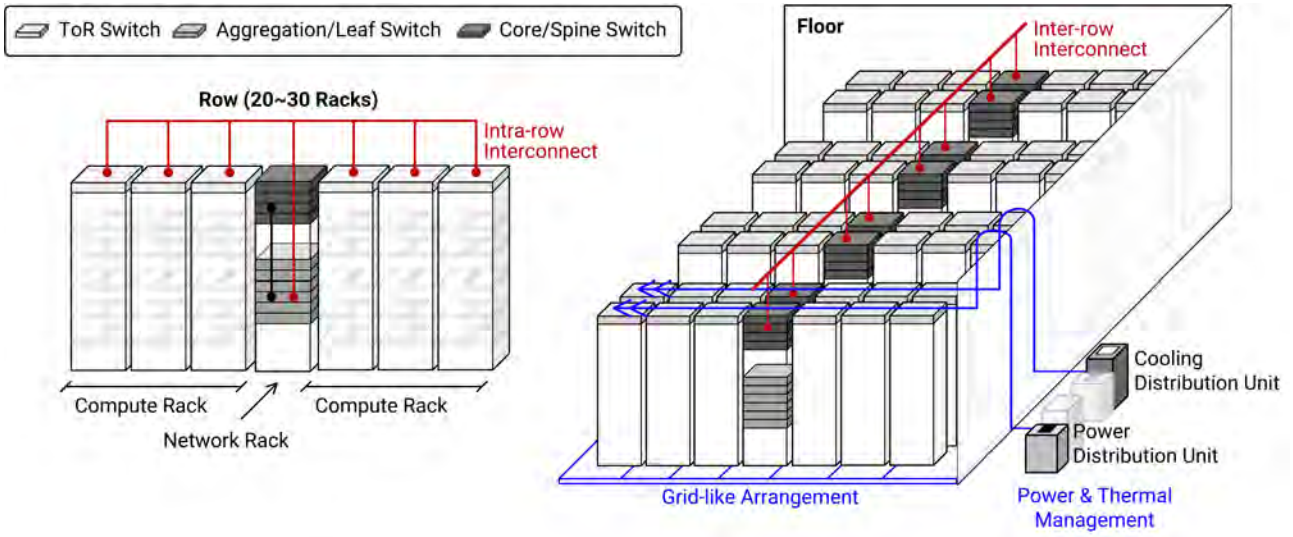
|그림 18| GB200 기반의 랙 수준 구성.

의 LPDDR5X DRAM을 갖추고 있고 [265–267], NVLink C2C 인터페이스를 통해 GPU와의 메모리 일관성을 유지하면서 약 900GB/s의 대역폭으로 통신한다 [37, 231, 232, 268–270]. 이 고집적 설계를 통해 GB200 모듈 내부의 CPU와 GPU는 노드 수준에서 일관된 공유 메모리 공간을 형성한다.

각 컴퓨트 노드는 데이터센터 외부의 네트워크 패브릭과 고속 연결을 위해 고성능 NIC을 내장한다. 고성능 통신을 위해 NVIDIA의 블루필드 데이터 프로세싱 유닛(DPU, Data Processing Unit [271, 272]) 또는 커넥트엑스(ConnectX) 어댑터 [273, 274]가 직접 장착되어 하드웨어 기반 네트워크 기능 가속을 지원하기도 한다. 일반적으로 노드당 400–800Gb/s 수준의 대역폭을 제공하는데 각 노드는 RDMA를 통해 저지연, 고 처리량 데이터 전송이 가능하다. 이렇게 구성된 노드들은 데이터센터 수준의 네트워크에 직접 참여하며 내부 클러스터 작업뿐 아니라 외부와의 통신도 처리할 수 있다. 예를 들어 그림 17b와 같이, 하나의 컴퓨트 노드는 두 개의 GB200 모듈(총 CPU 2개와 GPU 4개)을 탑재할 수 있으며, 이는 고밀도 랙 배치를 고려해 1U 또는 2U 폼팩터로 구성된다 [262, 263].

나아가 랙 수준 아키텍처에서는 이러한 다수의 컴퓨트 노드들을 집적하여 고밀도 연산 클러스터를 구성한다. 그림 18a는 GB200 기반의 대표적인 랙 아키텍처를 나타낸다. 이 아키텍처에서는 노드들이 NVLink나 UALink와 같은 내부 인터커넥트 패브릭을 통해 상호 연결된다. 일반적으로 단일 랙은 최대 36개의 GB200 모듈을 포함할 수 있으며, 이에 따라 총 72개의 GPU와 36개의 CPU로 구성되게 된다 [262, 263]. 랙 내부의 GPU들은 전용 내부 인터커넥트 스위치(NVSwitch 등 [71–73])를 통해 서로 연결되어 높은 대역폭과 낮은 지연을 제공할 수 있도록 설계될 수 있으며, 이는 빈번한 데이터 교환이 요구되는 대규모 모델 학습 및 실시간 추론에 적합한 "스케일업 도메인"을 형성하게 한다.

동시에, 그림 18b에서 볼 수 있듯이, 랙 내 각 노드는 "랙 상단(ToR, Top of Rack)"의 장거리 네트워크 스위치에 직접 연결된다. ToR 스위치 [275–278]는 랙 내 모든 노드의 트래픽을 집계하며 고성능 외부 대역폭을 동시에 관리할 수 있도록 해준다. 이를 통해 노드 간 네트워크 트래픽 관리가 용이해지고, 케이블 구성을 간소화할 수 있으며, 외부 통신 지연을 최소화하도록 설계한다. 결과적으로 이들을 어떻게 구성하는가에 따라 운영 효율성과 시스템 확장성이 완전히 달라질 수 있다. 참고로 데이터센터 인프라가 랙 단위를 넘어 확장될 수 있도록, ToR 스위치는 상위



(a) 열 수준 구성. **(b)** 층 수준 구성.

|그림 19| 열 수준과 층 수준 구성.

네트워크 장비와의 업링크 연결을 제공하도록 설계 및 구현되는 것이 일반적이다. 이러한 업링크는 스파인-리프 네트워크 구조에서 집계 스위치(Aggregation Switch [279, 280]) 또는 리프 스위치(Leaf Switch [281–284])에 연결되며, 구조화된 연결을 통해 랙 간, 열 간, 층 간 및 건물 간 확장을 가능하게 한다. 다시 말해 ToR 스위치는 랙 내부의 통신을 집계하는 동시에, 데이터센터 전체 네트워크와의 브리지 역할을 수행한다고 볼 수 있다. NVLink와 같은 랙 내부 인터커넥트와 ToR 스위치를 통한 외부 네트워크 연결의 결합은, 현대 AI 인프라 설계에서 GPU 성능과 확장성을 동시에 충족시키는 핵심 구조이자 확장성의 한계를 가져다주는 문제 요소이기도 하다.

열 및 층 수준의 구성: 인프라 확장. 현대 데이터센터 아키텍처는 일반적으로 고밀도의 컴퓨트 랙(Compute Rack)들을 여러 개의 열 단위로 구성하고, 각 열 내에서 발생하는 데이터 트래픽을 집계하고 라우팅하는 별도의 "네트워크 랙(Network Rack)"을 해당 열 내에 배치하는 방식으로 이루어진다. 네트워크 랙은 주로 ToR 스위치들이 연결되는 집계 스위치 또는 스파인-리프 구조의 스위치들로 구성되며, ToR스위치들과 함께 동일 열에 속한 여러 컴퓨트 랙 간 통신의 백본(Backbone)을 형성하는 데 사용된다.

그림 19a는 이러한 현대 데이터센터의 열 단위 다중 가속기 시스템 구성 사례로서, 복수의 컴퓨트 랙이 앞서 언급된 인피니밴드나 이더넷 기반의 고속 스위치를 통해 네트워크 랙과 연결된 구조를 보여준다 [285, 286]. 그림에서 볼 수 있듯, 각 열은 여러 개의 고밀도 컴퓨트 랙으로 구성되며, 각 랙 내부는 다수의 컴퓨트 노드들을 포함한다. 열 중앙이나 양단에 위치한 전용 네트워크 랙은 모든 컴퓨트 랙 간 데이터 트래픽의 집계 및 라우팅을 수행하도록 구성되어 있다. 예를 들어, 네트워크 랙의 스위치는 인피니밴드 기반의 퀀텀-2(Quantum-2 [287, 288])나 이더넷 기반의 스펙트럼-X(Spectrum-X [289])와 같은 고대역폭 구조를 사용하여 랙 간의 효율적이고 빠른 데이터 교환을 가능하게 한다.

네트워크 랙에 스위칭 인프라를 집중적으로 배치하는 이유로는 케이블 관리를 용이하게 하고 네트워크 지연을 감소시키며, 컴퓨트 랙을 추가하거나 확장할 때 유연성을 확보하기 위해서이다. 이렇게 구성된 집계 스위치 기반의

네트워크 랙 구조는 데이터센터의 운영 효율성과 확장성을 확보하는 데 효과적으로 알려져 있다.

이러한 열 단위의 통신 구조는 이제까지 노드라는 유닛과 랙이라는 유닛을 통 다중 가속기 시스템 규모를 확장하여 데이터센터를 구성하는 데 많은 도움이 되었다. 하지만 LLM과 같은 큰 모델과 데이터를 처리하는 데 있어서 최근 운영 환경에서는 GPU 간 고속 동기화 등에 다양한 구조적 한계를 만들어내고 있다. 특히, 앞서 언급된 GPU 간 빈번한 동기화 요구로 인해 GPU 활용률이 크게 저하될 수 있으며, 이에 따라 열 수준에서의 전략적 인프라 설계를 대규모 AI 워크로드의 안정성과 성능을 보장할 수 있도록 더욱 효율적으로 할 필요가 있다. 본 기술 보고서에서는 현재 스케일아웃 구조로 관리되고 있는 열 내부와 열 간 통신 모두의 중요성을 강조하며, 향후 스케일업 구조를 통해 이를 더 개선할 가능성을 함께 제시하고자 한다.

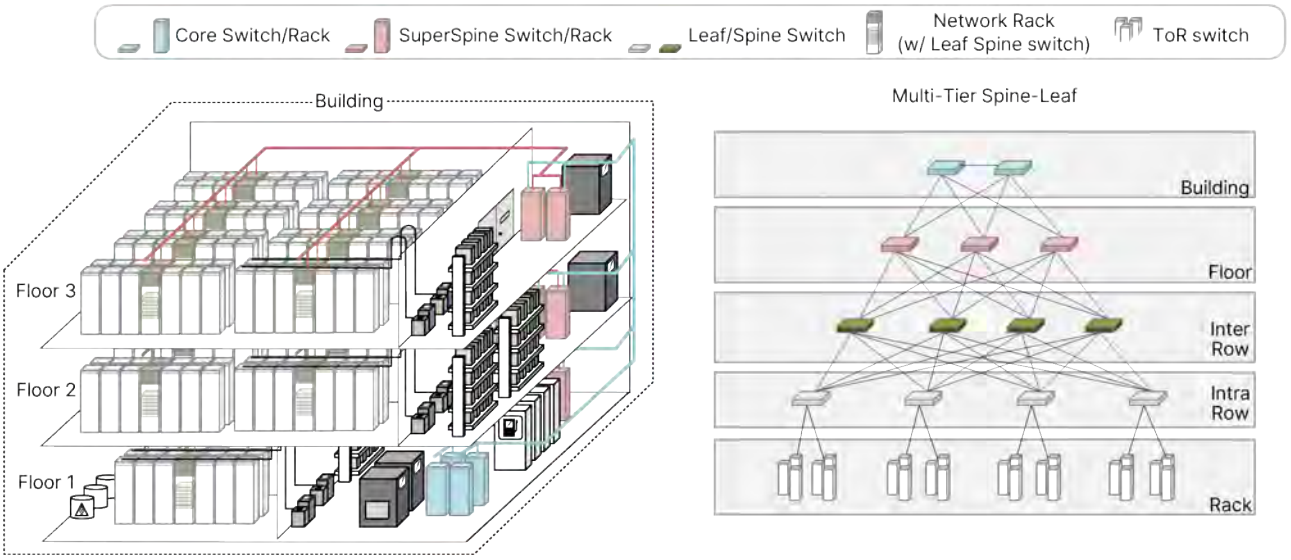
본 기술 보고서는 현재 AI 인프라에서 스케일아웃 방식으로 관리되는 열 내부 및 열 간 통신의 중요성을 강조하며, 향후 이러한 구조를 스케일업 방식으로 전환함으로써 추가적인 성능 향상이 가능함을 제시한다.

한편, 데이터센터 인프라 확장에서는 열 수준에서 층 수준으로의 전환이 핵심적 단계이다. 열 내부의 최적화는 층 단위 아키텍처로 자연스럽게 연결되어야 한다. 일반적으로 층 수준 아키텍처는 여러 개의 열을 그리드(Grid) 형태로 연결하여 공간 효율성과 열 간 효율적 통신을 동시에 달성할 수 있도록 설계된다 [256, 290]. 구체적으로, 하나의 층은 보통 20개에서 30개의 랙으로 구성된 여러 열을 포함하며, 이는 전체적으로 수백 개의 랙을 통합 관리하는 규모로 확장된다. 그림 19b는 이러한 층 단위 구성을 시각적으로 나타낸 것으로, 열 간 상호 연결 구조를 강조하며 효율적 데이터 흐름과 자원 공유를 가능하게 하는 배치의 예를 보여준다.

이 정도의 규모에서는 물리적 공간 구성, 효율적인 열 간 네트워크 연결, 그리고 인프라의 전반적인 체계적 관리가 매우 중요하다. 또한, 사용되는 인터커넥트 및 네트워크 기술의 특성에 따라 스케일업 및 스케일아웃 도메인의 설계 고려도 함께 이루어져야 한다. 데이터센터의 하드웨어 구성 특성상 열 간(Inter-Row) 및 열 내(Intra-Row) 통신은 필수적이며 빈번하게 발생하는데 이는 GPU 및 기타 가속기의 병렬 처리 동기화나 메모리 접근 시 필요한 데이터 교환 등이 주로 열 단위 내에서 이루어지기 때문이다. 그러나 현재는 대부분의 이러한 열 단위 통신이 스케일업이 아닌 스케일아웃 도메인을 통해 이루어져, 데이터 이동 시 상대적으로 높은 비용을 유발하고 있다. 우리는 향후 스케일업 아키텍처의 개선을 통해 열 내 및 열 간 연결을 저지연 고속 통신이 가능하게 하는 다양한 전략을 논의할 것이다.

이와 더불어, 고밀도 컴퓨팅 노드에서 발생하는 열을 효과적으로 관리하기 위한 고급 열 관리 및 전력 분배 솔루션 역시 중요하다. 예컨대, 액체 냉각 분배 장치(Liquid-Cooling Distribution Units [291, 292])나 전력 분배 장치(Power Distribution Units [260, 293, 294])는 효과적인 열 분산을 통해 시스템의 안정적인 운영 환경을 유지하고 성능 저하를 방지하는 데 결정적인 역할을 한다. 이러한 설계 전략은 궁극적으로 대규모 AI 인프라의 신뢰성과 효율성을 확보하기 위한 필수 요소 중 하나이다.

건물 수준의 통합: AI 인프라 구성에서 캠퍼스 규모로의 확장. 그림 20a에서 나타낸 것처럼, 건물 수준(Building-



(a) 건물 수준 구성.

(b) 데이터센터의 다단계 스파인-리프 토폴로지.

|그림 20| 건물 수준 구성과 전체 데이터센터 토폴로지.

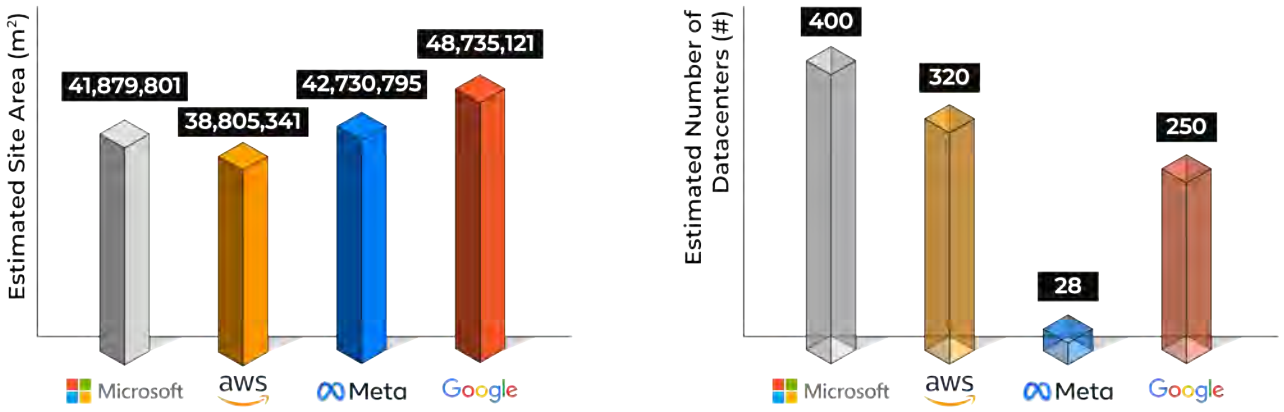
Level)의 통합은 계층적 데이터센터 아키텍처에서 최상위 계층을 구성하며, 여러 층을 상호 연결하여 수천에서 수만 대에 달하는 GPU 자원을 하나의 대규모 AI 인프라로 통합하는 것을 의미한다. 각 층은 다시 여러 열과 랙으로 구성되고, 이들 계층은 하나의 통합된 데이터센터 시스템으로 운영된다.

이러한 대규모 연산 환경에서는 정말 많은 수의 GPU가 여러 층에 걸쳐 배치되므로, 자원 할당의 일관성, 데이터 이동, 네트워크 조정과 같은 복잡한 운영 과제가 동반된다. 예컨대 현대의 대규모 배치에서는 여러 층에 위치한 GPU 클러스터를 장거리 네트워크 기술로 연결함으로써 건물 전체 수준에서 GPU 자원을 논리적으로 통합하고 있다 [295–297].

그러나 건물 단위로 인프라를 확장하면, 하위 계층(노드, 랙, 열)에서는 발생하지 않았던 새로운 문제들이 발생한다. 구체적으로, 층 간 통신으로 인해 새로운 형태의 네트워크 지연과 혼잡(Congestion)이 증가하며, 이는 GPU의 실질적인 활용률을 이론적 최대치의 절반 수준으로 제한하는 경우가 많다 [34, 255, 298]. 따라서 일반적으로 건물 수준의 통합에서는 다단계(Multi-Tier) 스파인-리프 또는 다단계 팻트리(Fat-Tree) 아키텍처와 같은 계층형 네트워크 토폴로지를 채택하여 통신 부하를 분산하고, 지연을 완화하며, 혼잡을 효과적으로 관리한다(그림 20b 참조).

이와 동시에, 전력 분배, 열 환경 제어(Thermal Regulation), 장애 허용성(Fault Tolerance) 등의 운영 문제도 규모가 커짐에 따라 다른 양상으로 발현하여 데이터센터 운영의 복잡성을 더욱 증가시킨다 [299, 300]. 이러한 문제들을 완화하기 위해서는 계산 부하, 네트워크 상태, 열 분산 상태, 하드웨어 상태 등에 대한 실시간 시각화와 이를 기반으로 하는 자동화된 모니터링 및 중앙 집중형 자원 관리 시스템이 필요하다.

그러나 이와 같은 구체적 운영 전략에도 불구하고 GPU 간 동기화 오버헤드와 계층 간 대용량 데이터 이동과 같은 본질적인 통신 병목 구조는 여전히 피할 수 없는 구조적 제약으로 남아 있다. 이러한 통신 병목은 결과적으로 시스템의 확장성과 효율성을 근본적으로 제한하게 되며, 이를 해결하기 위해 기존과 다른 형태의 인터커넥트 아키텍처와



(a) 하이퍼스케일러별 미국 내 데이터센터 면적.

(b) 하이퍼스케일러별 전체 데이터센터 수.

|그림 21| 하이퍼스케일러의 데이터센터 규모.

컴포저블 시스템 설계가 요구된다. 이에 대해서는 이후 섹션에서 보다 자세히 다룰 예정이다.

또한, 여러 건물 단위의 구조물이 결합하면 하나의 캠퍼스 규모 인프라(Campus-Scale Infrastructure)를 형성하게 되며, 이는 대규모 데이터센터 배치를 가능하게 한다. 그림 21a와 21b는 마이크로소프트(Microsoft), 메타(Meta), 구글(Google), 아마존(Amazon) 등 주요 하이퍼스케일러(Hyperscaler)가 구축한 데이터센터의 규모와 수량을 시각적으로 나타낸다. 이들 기업은 AI 인프라 수요의 급증에 대응하기 위해 데이터센터를 빠르게 확장하고 있으며, 이러한 확장은 설계 방식과 운영 구조 전반에 영향을 주고 있다.

그림 21a는 미국 내에서 운영 중이거나 2027년까지 완공이 예정된 각 기업의 데이터센터 부지 면적을 보여준다 [301]. 그림 21b는 각 기업이 자체 기준에 따라 정의한 데이터센터의 개수를 비교한 것이다 [302–306].

기업별 인프라 규모는 다음과 같다. 메타의 데이터센터 전체 부지 면적은 약 4,200만 m^2로, 이는 표준 축구장 약 5,300개에 해당한다. 마이크로소프트는 전 세계적으로 약 400개의 데이터센터를 운영 중이며, 아마존(AWS)와 구글은 각각 200∼300개 사이의 시설을 보유하고 있다. 메타는 약 30개 수준의 센터만 운영하지만, 각 시설의 면적이 매우 넓어 전체 부지 규모는 타 기업들과 유사한 수준이다. 메타의 인프라는 대규모 고밀도 구조를 기반으로 하며, 용량 확보와 운영 효율을 우선시하는 방향으로 설계되어 있다. 이러한 차이는 각 기업이 채택한 인프라 확장 전략의 방향성을 보여주며, 결과적으로 데이터센터의 설계 복잡성과 자원 활용 효율성에 영향을 미친다. 참고로 국내에 전역에 수십개의 클러스터가 데이터센터가 전역에 등록되어 있지만, 세종에 네이버가 AI용 하이퍼스케일용으로 인프라를 구축하고 있는 것의 부지 면적이 약 294천 m^2(축구장 41개)것을 고려해보면 현재 국내 인프라는 아직 그 규모가 매우 작은 수준임을 알 수 있다.

3.4. GPU 중심 AI 인프라가 가진 제약과 도전 과제

연산을 넘어선 한계: GPU 병렬화가 근본적으로 직면한 제약들. 지금까지 노드 단위에서 건물 규모까지 확장 가능한 블랙웰 기반 계층형 데이터센터 아키텍처를 통해 수천에서 수만 개의 GPU를 통합하는 구조적 확장성을 살펴보았다. 그러나 이 같은 아키텍처적 확장성에도 불구하고, GPU는 여전히 메모리 제한과 필연적인 통신 오버헤드로 인해 완벽한 연산 병렬화가 불가능하다.

앞서 논의한 바와 같이, 현대의 LLM 워크로드는 어텐션 메커니즘에서 생성되는 대규모 중간 상태(Intermediate State), 활성화값, 그리고 모델 파라미터의 규모로 인해 그 데이터 사이즈가 단일 GPU의 메모리가 수용할 수 있는 용량을 가뿐히 넘어선다. 따라서 다수의 GPU에 걸친 분산 연산(Partitioning)이 필수적이지만, 이러한 병렬화 과정에서는 GPU 간의 동기화 및 통신 오버헤드, 그리고 운영의 복잡성과 같은 중요한 성능 트레이드오프가 필연적으로 발생한다.

예를 들어, 모델 병렬화(Model Parallelism)는 모델 파라미터를 여러 GPU에 분산하여 메모리 용량 부족 문제를 해결할 수 있지만, GPU 간 빈번한 동기화 과정에서 큰 오버헤드를 발생시킨다. 반면 데이터 병렬화(Data Parallelism)는 각 GPU에 모델을 복제하고 배치 단위의 병렬 처리를 수행하지만, All-Reduce와 같은 집합 동기화 연산 때문에 GPU 활용률이 이론적으로 최대 35–40% 수준에 머무른다 [307–312].

앞서 논의 되었듯이 파이프라인 병렬화의 경우 모델을 여러 GPU에 계층적으로 분할하여 처리함으로써 더 큰 모델을 처리할 수 있지만, 계층 간 데이터 전송 과정에서 발생하는 파이프라인 버블로 인해 GPU 유휴 시간이 나타나며, GPU 활용률이 약 50% 수준으로 제한된다 [28, 45, 170, 207, 313]. 이러한 한계를 극복하기 위한 다양한 병렬화 기법을 결합한 하이브리드 병렬화(Hybrid Parallelism) 방식도 제안되었으나, 다중 GPU 환경에서 나타나는 본질적 통신 오버헤드를 근본적으로 해소하지는 못하고 있다.

중요한 점은 이러한 구조적 병목이 최신 하드웨어 아키텍처의 발전에도 불구하고 여전히 존재한다는 것이다. 고대역폭 NVLink 연결 기술, 확장된 HBM3e의 메모리 용량, 통합형 CPU-GPU 모듈 설계와 같은 하드웨어 혁신은 부분적으로 통신 지연을 완화하고 전체 처리량을 증가시킬 수 있지만, 근본적인 동기화 오버헤드와 메모리 관리의 복잡성 문제는 여전히 미해결 과제로 남아 있다. 결국, GPU 병렬화 기법이 본질적으로 가지고 있는 통신 오버헤드와 동기화 요구라는 구조적 제약은, 대규모 AI 인프라의 최적화와 아키텍처 발전을 위해 반드시 해결되어야 할 핵심 과제이다.

GPU 병렬화 기법이 가진 통신 오버헤드와 동기화 요구라는 구조적 제약은 대규모 AI 인프라의 성능 최적화와 아키텍처 발전을 위해 반드시 해결되어야 한다.

자원 통합 방식의 재고: 대규모 환경에서의 CPU-GPU 통합 아키텍처의 한계. 블랙웰 아키텍처와 같은 CPU-GPU 통합 모듈은 특정 연산이 집중된 워크로드에서 우수한 성능을 제공하도록 설계되었으나, 이러한 모듈을 수천에서

수만 개의 노드를 갖춘 대규모 데이터센터 전반에 배치할 경우, 확장성, 유연성, 운영 효율성 면에서 명확한 구조적 제약을 드러낸다.

무엇보다도 통합 모듈은 연산, 네트워크, 메모리 자원 간에 끊어 낼 수 없는 결합성이 근본적으로 존재하므로(Rigid Resource Coupling), 각 자원을 독립적으로 확장하는 데 필요한 유연성이 심각히 제한된다. 대규모 AI 환경에서는 이와 같은 결합 구조가 두 가지 근본적인 문제를 더욱 심화시킨다. 첫째, 노드 간 통신 오버헤드(Inter-Node Communication Overhead)의 선형적 증가이다. 각 CPU-GPU 노드는 다른 노드와의 데이터 교환 및 동기화를 위해 별도의 네트워크 연결을 필요로 하므로, 전체 네트워크 토폴로지가 복잡해지고 GPU 간 지연과 동기화 시간이 심각히 늘어나게 된다. 결과적으로 LLM의 학습 및 추론과 같은 통신 중심 워크로드에서 성능 저하가 발생할 수 밖에 없다.

둘째, 고정된 CPU:GPU 비율(예: GB200/300의 경우 1 CPU : 2 GPU)은 인프라 규모가 커질수록 CPU 자원의 과도한 프로비저닝(Over-Provisioning)을 초래하는 비효율성을 낳게 된다. 현재 LLM 구조상 연산의 대부분이 GPU 중심으로 이루어짐에 따라, CPU는 상대적으로 유휴 상태로 남는 시간이 많아져 자원이 낭비되고 비용이 자연스럽게 상승할 수 밖에 없다. 이와 함께, 각 노드에 물리적으로 고정된 메모리 바인딩(Binding)은, 메모리 그 자체만으로 확장하기 어려우므로 노드를 추가하면 메모리 용량도 어쩔 수 없이 비례적으로 증가해야 한다. 이는 특정 워크로드에서는 메모리의 낭비를 야기하고, 다른 경우에는 메모리가 부족하여 모델 크기나 배치 사이즈(Batch Size)를 제한하는 상황을 만들게 된다.

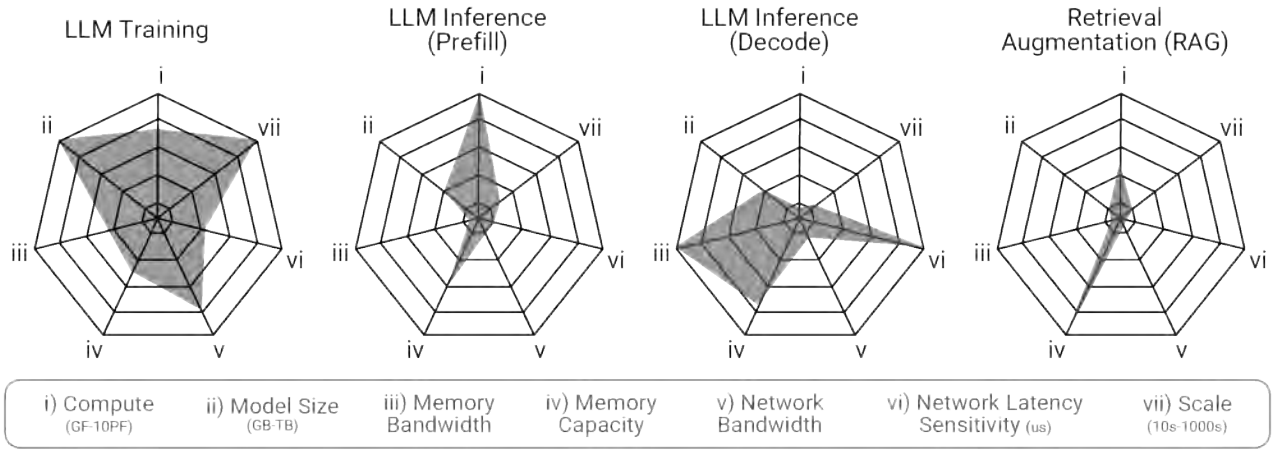
마지막으로, CPU-GPU 통합 모듈은 유지보수 및 업그레이드 과정에서 심각한 제약을 일으킨다. 모듈 내 하나의 구성 요소에 결함이 발생하면 전체 모듈을 교체해야 하므로, 시스템 다운타임(Downtime)과 운영 비용이 증가한다. 또한 CPU와 GPU를 개별적으로 업그레이드하는 것이 어렵기 때문에, 최신 기술 진보를 신속히 반영하기 힘들고, 이는 시스템 민첩성(Agility)과 현대화 속도를 떨어뜨린다.

NVIDIA의 블랙웰 아키텍처와 같은 CPU-GPU 통합 모듈의 구조적·운영적 제약은 AI 인프라에서 CPU, GPU, 메모리, 네트워크 자원을 모듈형으로, 그리고 독립적으로 확장할 수 있는 구조로 재설계하고 적용해야함을 보여준다

이러한 구조적·운영적 제약은 CPU-GPU 통합 모듈이 대규모 AI 인프라의 유연하고 확장 가능한 자원 요구를 충족하는 데 본질적인 한계를 가지고 있음을 명확히 보여준다. 따라서 이러한 병목을 해결하기 위해 향후 데이터센터 아키텍처는 CPU, GPU, 메모리, 네트워크 자원을 모듈형으로, 그리고 독립적으로 확장할 수 있는 구조로 재설계되어야 한다. 자원을 분리하여 구성하는 접근 방식만이 대규모 인프라 환경에서 확장성, 운영 유연성, 자원 활용 최적화를 동시에 달성할 수 있는 실질적인 해결책이 될 것이다 [251–254, 314].

4. 다양한 성능 지표 최적화를 위한 CXL 기반 모듈형 아키텍처

현대의 AI 인프라가 동시에 고려해야 하는 성능 지표를 그림 22에 나타내었다. 여기에는 연산 처리량(Computational Throughput), 모델 크기(Model Size), 메모리 대역폭(Memory Bandwidth), 메모리 용량(Memory Capacity), 네트워크 대역폭(Network Bandwidth), 지연 민감도(Latency Sensitivity), 전체 시스템 확장성(System Scalability)이 포함된다. 이러한 성능 지표의 상대적 중요도는 서비스 시나리오마다 다르게 나타나기 때문



|그림 22| 서비스 시나리오별 성능 지표의 상대적 중요도.

에, AI 인프라에 대한 구체적인 요구사항은 워크로드 특성에 따라 동적으로 변화할 수 있음을 알 수 있다 [315–319].

본 절에서는 LLM 학습 [7, 8, 320], LLM 추론 단계의 "사전 채움(Prefill)" 및 디코딩(Decode) [159, 321, 322], RAG 워크로드 [64, 323, 324]와 같이 서로 구별되는 AI 워크로드 시나리오를 앞서 언급한 성능 지표 중 서로 다른 조합으로 표현해보고 그 이유를 분석해본다. 이러한 분석은 단일 아키텍처 구성이 모든 서비스 시나리오에서 요구하는 다양한 성능 지표를 동시에 최적화할 수 없다는 점을 보여줄 것이다. 이와 같이 다차원적이며 지속적으로 진화하는 성능 요구들을 해결하려면, 연산, 메모리, 네트워크 등 주요 자원을 독립적으로 확장 가능한 모듈형 컴포저블 아키텍처가 필수적이다.

본 보고서는 컴퓨트 익스프레스 링크(CXL, Compute Express Link)를 통해 이러한 문제를 효과적으로 해결할 수 있음을 다방면으로 분석하고 보여주고자 한다. 기본적으로 CXL은 자원의 물리적 분리(Disaggregation)와 동적 재구성(Dynamic Composability)을 지원하여, 각 AI 워크로드 특성에 맞추어 인프라 자원을 유연하게 구성하고 효율적으로 확장할 수 있다.

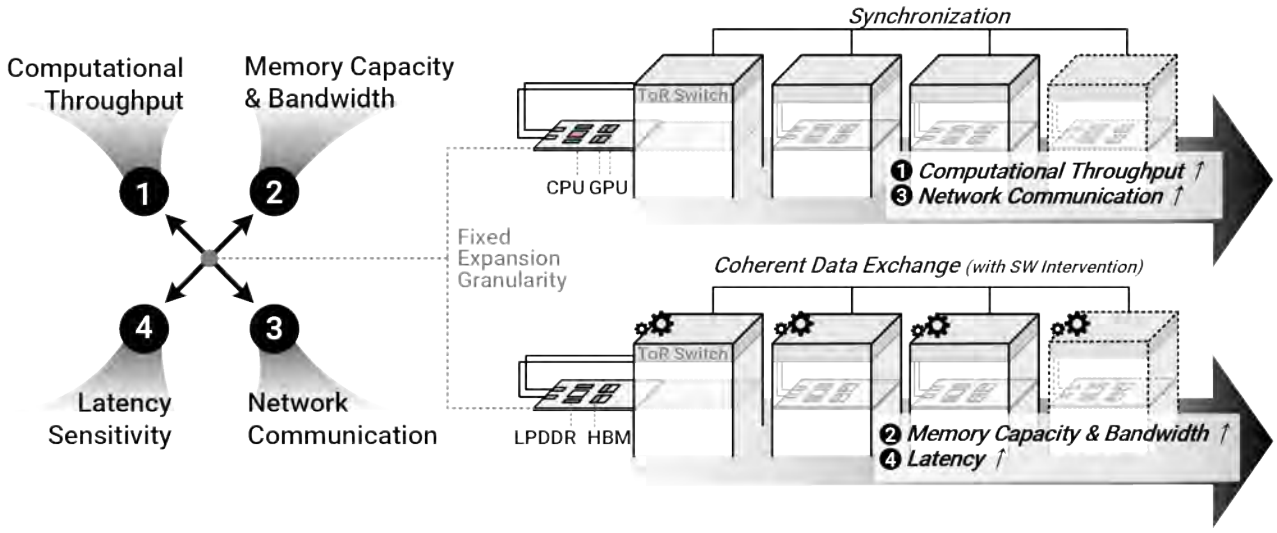
이 섹션에서는 우선 현대 AI 인프라가 직면한 다양한 성능 요구사항과 관련된 근본적 도전 과제들을 논의한다. 이어서 CXL 사양의 발전과 배경지식, 그리고 그 아키텍처적 의미를 분석하며, 이를 통해 다양한 AI 워크로드의 요구를 만족시키는 방법을 살펴본다. 마지막으로, 여러 워크로드 시나리오에서 성능 지표를 동시에 최적화할 수 있도록 설계한 CXL 기반의 트레이 디자인과 랙 아키텍처를 제안한다.

4.1. AI 워크로드 특성에 따른 성능 지표 분류 및 기존 아키텍처의 한계

현대 AI 인프라가 동시에 충족해야 하는 일곱 가지 성능 지표를 명확히 이해하기 위해, 우리는 각 지표에 영향을 미치는 요소들을 기계학습 워크로드 특성에 따라 (1) 연산 처리량, (2) 메모리 용량 및 대역폭, (3) 네트워크 통신, 그리고 (4) 지연 민감성 등 네 가지 주요 그룹으로 나누어 분류하였다. 본 절에서는 각 그룹이 기계학습 워크로드를 실행하는 데 있어서 왜 중요한지 알아보고, 기존의 데이터센터 아키텍처와 인프라가 이처럼 서로 밀접히 연결된 여러 성능 지표들을 동시에 최적화하기 어려운 이유에 대해 분석해 본다.

AI 인프라에서는 실행 응용과 워크로드에 따라서 요구하는 지표가 모두 달라지기 때문에 하나의 도메인 특화 장치나 가속기 설계로 이를 모두 만족하기 어렵다.

연산 처리량(Computational throughput). LLM은 충분한 표현력과 우수한 일반화 성능을 확보하기 위해 수백억에서 수천억 개의 파라미터로 이루어진 매우 큰 모델 크기를 요구한다. 특히 이러한 대규모 모델은 작은 규모의 모델이 얻기 어려운 복잡한 언어 관계와 미묘한 문맥적 차이를 효과적으로 포착할 수 있으며, 이를 통해 고급 추론(Advanced Reasoning)과 향상된 문맥 이해(Improved Context Comprehension)와 같은 이머전트



(a) 성능 지표 간 상충 관계. **(b)** 성능 지표 간 상충 예시.

|그림 23| 성능 지표 간 상충 관계와 그 예시.

(Emergent) 능력을 나타낸다. 예를 들어 MoE 구조를 채택한 모델은 각 입력 토큰에 따라 특정 전문가 네트워크를 선택적으로 활성화하여 파라미터 수를 종종 수조 개 수준으로까지 증가시키고, 모델의 용량과 성능을 크게 향상한다.

이러한 이유로 방대한 파라미터 집합을 현실적인 시간 내에 처리하기 위해서는 연산 처리량이 매우 중요하고, 이러한 높은 연산 처리량을 얻기 위해서는 수천 개 이상의 GPU를 이용한 병렬 처리가 필수적이다. 그러나 앞서 3.4절에서 논의된 것처럼, 대규모 GPU 클러스터 운영 시 발생하는 빈번한 동기화 이벤트는 근본적으로 제거될 수 없다. 특히 그래디언트 집계 및 MoE 구조 특유의 전문가 네트워크 활성화 등을 수행하는 데 발생하는 집합적 연산 과정에서 GPU 간 높은 빈도의 동기화가 필요하다. 이러한 집약적인 집합 통신 패턴은 네트워크 대역폭 수요를 크게 증가시켜 필연적으로 성능 병목을 초래한다. 결과적으로 동기화로 인한 오버헤드와 네트워크 혼잡으로 인해 실제 GPU 활용률은 이론적으로 달성 가능한 최대 성능의 절반에도 미치지 못하는 수준에 그친다 [28, 99, 100, 170, 207, 309, 325, 326].

메모리 용량과 대역폭(Memory capacity and bandwidth). 메모리 용량과 대역폭 또한 연산 처리량 못지않게 중요한 그룹 요소이자 병목 요인이다. 최근의 대규모 LLM은 모델 파라미터 및 중간 활성화값(Intermediate Activation)의 수가 수백억에서 수천억 개 규모로 증가하면서, 학습 과정에서 필요한 메모리 용량이 수백 테라바이트를 넘어섰다. 실제 학습 환경에서는 모델 파라미터뿐 아니라 이러한 중간 활성화값, 옵티마이저 상태(Optimizer State), 그래디언트 버퍼(Gradient Buffer), 그리고 메타데이터(Metadata) 등도 동시에 메모리에 적재해야 하므로 총 메모리 요구량은 더욱 증가한다 [24, 25, 327].

LLM의 방대한 메모리 요구량은 현재 아키텍처에서 GPU당 제공하는 로컬 메모리 용량(최대 수백GB 수준)을 훨씬 초과한다. 따라서 CPU와 GPU가 고정된 메모리 구성으로 밀접히 결합된 기존 아키텍처는 메모리 자원을 독립적으로 확장하는 데 유연성이 부족하여 수백 테라바이트 이상의 대규모 메모리 요구사항을 효과적으로 수용하기 어렵다. 또한 키-값 캐싱 및 RAG와 같이 메모리를 집중적으로 사용하는 최적화 기법들은 대규모 메모리 트랜잭션을 자주 유발하여 메모리 대역폭 제약을 더욱 심화시킨다. 따라서 연산 처리량 중심의 기존 아키텍처는 고대역폭의 지속적인

메모리 트래픽을 효율적으로 지원하지 못해, 성능 저하를 초래하는 한계를 보인다.

네트워크 통신(Network communication). 효율적인 네트워크 통신은 워크로드가 여러 노드에 걸쳐 분산되는 환경에서 필수적이며, 효과적인 병렬화 전략이 필요하다. 기존의 밀접히 결합되는 아키텍처는 주로 노드 내부(Intra-Node) 및 랙 내부(Intra-Rack) 연산 성능 최적화에 집중하는 반면, 노드 간(Inter-Node)과 랙 간(Inter-Rack)의 대규모 통신 요구를 충분히 고려하지 않는 경향이 있다.

병렬화 범위가 여러 노드로 확장될 경우, 노드 간 통신 빈도와 데이터 크기는 급격히 증가하여, 실제 GPU 내 데이터 크기보다 수 배에서 수십 배 큰 데이터가 교환될 수 있다. 특히, 대규모 LLM 학습 과정에서는 반복마다 GPU들이 중간 활성화값, 옵티마이저 상태, 그래디언트 업데이트 등 수백 TB 규모의 데이터를 관리한다. 어텐션 벡터, 그래디언트, KV 캐시의 빈번한 동기화로 인한 GPU 간 통신량은 페타바이트(PB) 수준까지 증가하여, 랙 단위에 존재하는 모든 GPU의 메모리 수용량을 넘어서는 규모가 되기도 한다.

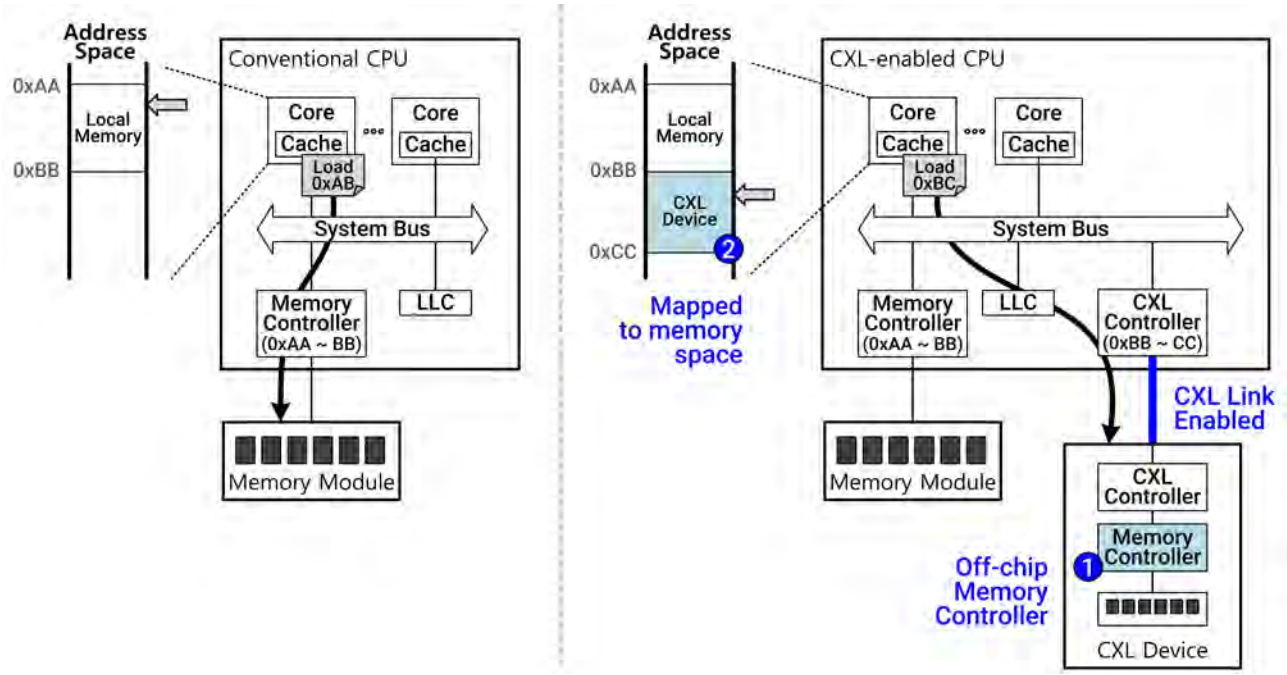
이러한 LLM의 대규모화는, 기존 서비스와 달리 데이터센터 아키텍처의 네트워크 병목 현상을 심각히 가속화하였으며, GPU 간 효율적인 동기화를 저해하고 데이터센터 수준의 모델 확장을 어렵게 하고 있다. 결론적으로 이러한 통신 오버헤드와 도전 과제는 확장 가능한 고대역폭·저지연 인터커넥트 인프라의 중요성을 다시 한번 강조한다.

지연 민감도(Latency sensitivity). 지연 민감도는 추론 과정에 전반적으로 중요하지만 특히 오토레그레시브 추론 과정의 디코딩 단계에서 핵심 설계 요소이다.

실시간 AI 워크로드는 GPU 간 중간 연산 결과의 빠르고 정확한 동기화를 통한 즉각적인 응답을 요구한다. 그러나 기존 인프라 구조는 노드 간의 빈번한 데이터 이동과 이더넷(Ethernet)이나 인피니밴드(InfiniBand) 같은 네트워크 기반 통신 기술에서 발생하는 소프트웨어 오버헤드로 인해 상당한 지연을 초래한다. 구체적으로 운영체제상의 권한 모드 전환(커널 모드와 유저 모드 간 전환), 불필요한 메모리 복사, 인터럽트 처리 및 프로토콜 처리 등과 같은 소프트웨어 간섭과 불필요한 프로세스가 높은 지연을 유발한다. 이러한 네트워크 오버헤드는 CXL이나 UALink와 같이 소프트웨어 개입이 없는 하드웨어 인터커넥트에 비해 수십에서 수백 배 높은 지연을 초래하여, 시스템 성능과 확장성을 크게 제한한다. 결과적으로 기존 아키텍처는 실시간 응답 요구를 충족하지 못하며, 지연에 민감한 AI 추론 작업의 활용성을 제한하게 된다.

앞서 언급한 네 가지 성능 지표, 즉 연산 처리량, 메모리 용량 및 대역폭, 네트워크 통신, 지연 민감성은 서로 상충하여 동시에 최적화하기 어렵다(그림 23a 참조). 예를 들어, 그림 23b에 보여지듯, GPU 병렬화를 통한 연산 처리량 증가는 네트워크 대역폭 수요를 증가시키고, 동기화 오버헤드를 높인다. 대규모 파라미터와 활성화 데이터를 위한 메모리 확장 역시 노드 간의 빈번한 캐시 일관성 데이터 교환을 초래하여 지연과 복잡성을 증가시킨다. 따라서 CPU와 GPU를 밀접하게 결합한 기존 아키텍처는 개별적인 자원 확장 유연성이 부족하여 자원 활용의 비효율성과 성능 제한을 초래할 수밖에 없다.

이러한 구조적 한계를 극복하기 위해, 차세대 AI 데이터센터는 연산, 메모리, 네트워크 자원을 독립적으로 모듈화하고, 개별적으로 확장 가능한 컴포저블 아키텍처를 채택해야 한다. 이러한 맥락에서 자원 분리, 동적 조합 가능성, 일관성 보장 메모리 풀과 같은 고급 기능을 제공하는 CXL이 현실적인 대안으로 부상하고 있다. 특히 CXL은 메모리 자원을 물리적으로 연산 장치에서 분리시키는 동시에 캐시 일관성을 유지하고 지연과 동기화 오버헤드를 낮추어 전체 시스템의 효율성을 향상시킬 수 있다.



|그림 24| CXL 기반의 CPU와 메모리 컨트롤러 분리 구조.

4.2. 컴포저블 아키텍처의 진화: CXL 기술의 발전

CPU로부터 메모리 자원의 분리: CXL 1.0. 앞서 논의한 확장성, 유연성 및 성능 문제를 해결하기 위해, CXL의 다양한 인터커넥트 특성을 고려하여 데이터센터의 아키텍처를 재구성할 수 있다. 전통적인 컴퓨터 구조에서는 메모리 컨트롤러가 CPU 패키지 내부에 긴밀히 통합되어 있어, 메모리 용량의 확장이나 메모리 자원의 물리적 분리가 거의 불가능하였다. 동적으로 변하는 워크로드의 입출력 데이터 크기 및 패턴을 효과적으로 수용하고, 노드 간 불균형한 메모리 자원의 활용도를 높이며, TCO를 감소시키기 위해 CPU와 메모리 자원의 물리적 분리는 필수적이다. 그러나 기존 컴퓨터 구조의 한계로 인해 지금까지 데이터센터는 주로 RDMA 기술을 활용하여 메모리를 논리적으로만 분리하고, 소프트웨어의 지원을 받아 여러 연산장치들이 공유하는 방식을 채택해왔다 [79, 328–331]. 이러한 RDMA 방식은 하드웨어 구조상 유연성은 제공하지만, 운영체제 권한 모드 전환, 데이터 직렬화 및 역직렬화, 불필요한 메모리 복사와 같은 소프트웨어 오버헤드가 불가피하여 성능 열화가 크고, 추가적인 데이터 이동과 관리로 인해 에너지 소모가 심각하다는 단점을 갖는다.

CXL의 프로토콜 인터페이스는 메모리 컨트롤러를 CPU로부터 물리적 및 논리적으로 완전히 분리할 수 있도록 하여 기존 컴퓨터 구조를 보완하기 위한 RDMA나 다른 기타 소프트웨어 방식의 근본적인 문제를 직접 해결하였다. 그림 24에 나타난 것처럼, CXL은 메모리 컨트롤러를 CPU 패키지 내부가 아닌 외부 메모리 모듈에 배치하여, 이를 엔드포인트(Endpoint)라 불리는 DRAM 확장 카드나 특수 메모리 장치 형태로 구현할 수 있도록 하였다. CXL 프로토콜의 특성으로 인한 하드웨어 구조적 변화 가능성 때문에 메모리 자원은 전통적인 CPU의 제약에서 벗어나 독립적이고 조합 가능한 형태로 메모리 자원 분리가 가능해졌다. 특히 CXL은 기존 PCI 익스프레스(PCIe, Peripheral Component Interconnect Express)의 물리 계층을 유지하면서, CPU의 표준적인 로드/스토어 명령을 통해 접근

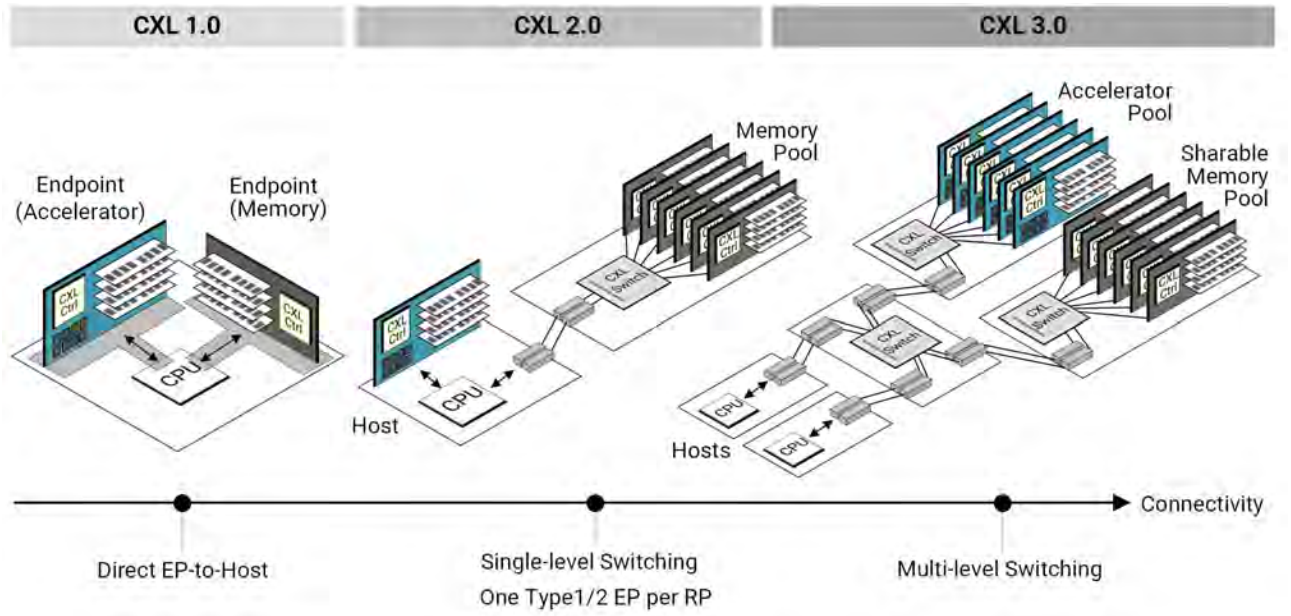
|표 1| CXL 버전별 비교 분석.

기능	CXL 1.0	CXL 2.0	CXL 3.0
최대 링크 속도 (GTs)	32	32	64
68바이트 플릿 (up to 32 GTs)	✓	✓	✓
256바이트 플릿 (up to 64 GTs)	–	–	✓
메모리 컨트롤러 분리	✓	✓	✓
메모리 확장	✓	✓	✓
메모리 풀링	–	✓	✓
메모리 공유	–	–	✓
단일 계층 스위칭	–	✓	✓
다단계 스위칭	–	–	✓
계층 기반 라우팅 (HBR)	–	✓	✓
포트 기반 라우팅 (PBR)	–	–	✓
핫플러그 지원	–	✓	✓
루트 포트당 최대 가속기 수	1	1	256
루트 포트당 최대 메모리 확장 장치 수	1	256	4096
백-인밸리데이션	–	–	✓
점대점(P2P, Peer-to-Peer) 통신	–	–	✓
출시 연도	2019	2020	2022-23

가능하고 캐시 일관성이 보장되는 메모리 인터페이스를 제공한다. 이를 통해 컨텍스트 스위칭이나 불필요한 메모리 복사 같은 소프트웨어 오버헤드를 제거하고, 하드웨어 수준에서 직접적인 메모리 접근 경로를 확보하여 외부 메모리 확장치에 대한 지연 시간을 RDMA 등에 비해 현저히 단축한다. 또한, CXL은 기존 PCIe 인프라에 링크 계층과 트랜잭션 계층과 같은 새로운 논리 계층을 추가로 구현하여, 기존 하드웨어 생태계에 별도의 수정 없이 쉽게 통합할 수 있도록 설계되었다.

표 1에서는 CXL 1.0, 2.0 및 3.0의 주요 기능을 비교하고, 각 버전에서의 확장성, 연결성 및 고급 기능의 점진적 발전 과정을 정리하였다. 최초의 CXL 1.0 [54] 사양은 메모리 컨트롤러 분리의 핵심 개념을 처음으로 제안하여 조합 가능한 인프라 구축의 기반 프레임워크를 제공하였다. 그러나 실제 구현에서 CXL 1.0의 확장성은 여전히 노드 내부로 제한되었다. 각 CXL 엔드포인트에 있는 외부 메모리 컨트롤러는 신호 무결성(Signal Integrity) 및 신호 감쇠(Signal Attenuation) 같은 물리적 한계 [332, 333]와 제한된 메모리 채널 수로 인해, 보통 엔드포인트당 최대 1TB에서 2TB 수준의 메모리 용량만 제공할 수 있다. 노드 내부에 적용할 수 있는 엔드포인트의 수는 물리적으로 한정되어 있으므로 보다 광범위하고 효율적인 메모리 자원의 분리를 실현하기 위해서는 추가적인 구조적 개선이 필요했고, 이는 이후 등장한 CXL 2.0 등 후속 버전 개발의 필요성을 촉진하였다.

스위치 기반 아키텍처를 통한 확장 가능한 컴포저블 메모리: CXL 2.0. 메모리 용량, 엔드포인트 확장성, 그리고 물리적으로 고정된 연결 방식과 같은 CXL 1.0의 여러 가지 한계를 극복하기 위해 CXL 2.0 [55]에서는 전용 스위치를 기반으로 하는 토폴로지를 도입하였다. 기존 CXL 1.0은 엔드포인트와 호스트 간의 직접 연결을 통해 제한된 확장성을 가졌던 반면, CXL 2.0에서는 중간에 스위치를 배치하여 다양한 메모리 자원을 더욱 유연하게 통합 및 관리할 수 있도록 하였다. 그림 25에서 나타낸 것과 같이, 컴퓨트 노드와 메모리 엔드포인트 간 스위칭 계층이 추가됨으로써,



|그림 25| 직접 연결에서 다단계 스위칭까지의 CXL 발전 과정.

호스트는 다수의 외부 엔드포인트들로 구성된, 확장성이 높고 더 큰 메모리 풀에 접근할 수 있게 되었고, 기존 점대점(P2P, Point-to-Point) 연결 방식에서 발생하던 연결성 병목을 효과적으로 해소할 수 있게 되었다.

내부적으로 CXL 2.0 스위치는 고대역폭의 크로스바(Crossbar) 아키텍처를 사용하여, 연결된 여러 엔드포인트와 컴퓨트 노드 간의 일관된 메모리 트랜잭션(Transaction)을 라우팅할 수 있다. 이러한 하드웨어 중심의 메모리 일관성 통신 방식은 대용량의 외부 메모리에 접근하는 데 기존 RDMA 네트워크 기반의 연결 방식에서 나타났던 소프트웨어 오버헤드를 제거하여 지연을 크게 감소시킨다. 또한, PCIe Gen5 기술(32 GT/s per lane)을 사용하는 CXL 2.0 스위치는 16레인(Lane) 기준 양방향 최대 64 GB/s의 대역폭을 제공한다. 결과적으로, 단일 CXL 2.0 스위치는 노드당 수십 TB 이상의 메모리를 통합할 수 있고 다수의 노드를 연결할 수 있으므로, 기존 엔드포인트 중심의 CXL 1.0 설계가 갖는 확장성의 한계를 뛰어넘을 수 있다. 이와 같은 확장성이 높은 스위치 토폴로지는 모듈식 시스템 증설을 보장하며 자원 프로비저닝을 단순화하고, 엔드포인트 중심의 제약에서 메모리 장치의 할당을 비교적 자유롭게 한다. 또한 CXL 2.0에서 새롭게 도입된 핫플러그(Hot-Plug [58]) 기능은 최소한의 운영 중단으로 메모리 엔드포인트를 동적으로 추가하거나 제거할 수 있도록 지원한다. 호스트 단위의 정적 메모리 할당 기능 [58, 60, 334] 역시 변화하는 워크로드의 요구사항에 효율적으로 대응하여 데이터센터의 운영 유연성을 높일 수 있게 하여 다양한 방면에서 CXL 1.0보다 기술적 우월성을 확보하였다.

CXL 2.0은 단일 계층 스위치만을 지원하여 유연성과 확장성이 제약되며 매우 소수의 GPU나 가속기 유형의 장치만을 연결 할 수 있어서 그 한계가 존재한다.

그러나 이러한 개선에도 불구하고, CXL 2.0은 여전히 대규모 시스템 구축을 지원하는 데 제한 요소를 가지고 있다. 특히, 계층적 다단계 스위치(Hierarchical Multi-Level Switch) 구성을 지원하지 않아 시스템의 구성이 단일 계층 스위치(Single-Layer Switch) 구조로 제한된다. 이는 메모리 풀의 규모와 루트 포트(Root Port)당 연결 가능한 장치 수를 크게 제한하는 결과를 가져왔다. 일반적으로 CPU는 제한된 수의 루트 포트를 제공하며, 각 포트는 고정된 연결 방식과 엄격한 대역폭 제약을 갖고 있다. 따라서 실제 환경에서는 CXL 2.0의 루트 포트당 메모리 확장 엔드포인트(Type 3 장치)는 보통 4개에서 16개 정도로 제한되었으며, 이론적으로 최대 수용가능한 256개 Type 3 장치와는 상당한 차이가 있었다. 또한, 가속기(Type 1 및 2 장치)의 경우 캐시 일관성을 유지하기 위해 포트당 한 개의 장치만 연결할 수 있기 때문에 배치의 유연성과 확장성이 더욱 제약되었다.

이러한 확장성의 제약은 이후의 CXL 3.0 사양에서 다단계 스위치 구성, 고급 라우팅 메커니즘 및 시스템 전반의 포괄적인 메모리 일관성 유지와 같은 추가적인 기술 발전을 만들어 내는 데 중요한 동기가 되었다.

다단계 스위칭과 메모리 공유를 통한 진정한 컴포저블 아키텍처: CXL 3.0. CXL 3.0 사양(Specification) [56][6]은 기존 CXL 2.0이 가진 확장성 및 계층적 구성의 한계를 극복하기 위해 구조적으로 많은 변화를 겪었다. 특히, 대규모 데이터센터와 고성능 컴퓨팅 환경에서의 진정한 컴포저블 아키텍처의 구성 및 자원 공유가 가능하도록 근본적인 설계 변화가 있었다. 기존의 CXL 2.0은 단일 계층의 스위치 토폴로지로 제한되어, 연결 가능한 엔드포인트의 수가 매우 제한적이었던 반면, CXL 3.0은 다단계 스위치 토폴로지인 "스위치 캐스케이딩(Cascading)"을 명시적으로 지원하여 인터커넥트로 패브릭을 구성, CXL 2.0이 가지고 있던 문제를 완전히 제거할 수 있게 하였다. 다시 말해 CXL 3.0은 여러 계층의 스위치를 상호 연결할 수 있고, CPU 루트 포트에서 일관성 있게 접근 가능한 엔드포인트(메모리 확장 장치 및 가속기)의 수를 대폭 늘렸다.

구체적으로, CXL 3.0 프로토콜 기술은 CPU 루트 포트당 최대 4,096개의 메모리 확장 장치(Type 3 장치) 연결을 지원하여, 대규모 AI 및 데이터 분석 워크로드에서 요구되는 대형 메모리 풀의 구축을 가능하게 한다. 가속기(Type 1 및 Type 2 장치) 통합 능력 또한 향상되어, 루트 포트당 최대 256개의 가속기를 지원하며 기존의 단일 장치 연결 한계를 크게 뛰어넘는다. 이러한 CXL의 아키텍처적 개선은 이기종 가속기 클러스터의 동적이고 유연한 배치를 실현하여, AI 인프라의 효율성을 최적화하고 변화하는 워크로드의 사용자 요구사항에 신속히 대응할 수 있게 한다.

CXL 3.0의 다단계 스위치 패브릭은 이전 CXL이 사용하던 계층 기반 라우팅(HBR, Hierarchy-Based Routing)을 보완하기 위해 새롭게 도입된 포트 기반 라우팅(PBR, Port-Based Routing)을 활용한다. 기존의 HBR은 계층적으로 고정된 경로와 정적 메모리 파티셔닝만 제공하기 때문에 자원의 동적 공유가 제한적일 수밖에 없었다. 이에 비해 PBR은 실시간 포트 상태와 네트워크 혼잡도에 따라 동적으로 최적 경로를 선택할 수 있다. 따라서 여러 호스트가 메모리 자원을 동시에 접근하며 진정한 다중 호스트(Multi-Host) 메모리 공유를 지원할 수 있다. 이는 결과적으로 트래픽 분산 효과를 높이고 지연을 낮추며, 통신 병목 현상을 완화함으로써 기존 CXL 2.0의 정적 방식이 가진 한계를 효과적으로 극복한다.

무엇보다, CXL 3.0은 이러한 고급 PBR 메커니즘을 통해 강력한 다중 호스트 메모리 공유와 시스템 전체의 포괄적인 캐시 일관성을 제공한다. 이러한 아키텍처적 개선은 대규모 AI 워크로드에서 연산 효율성을 높이고, 전체 시스템의 지연과 오버헤드를 최소화할 수 있게 한다. 예를 들어, 가속기와 컴퓨트 노드는 임베딩 테이블, KV 캐시 및 중간 활성화와 같은 주요 데이터 구조를 중복된 데이터 전송이나 RDMA 또는 캐시 일관성을 보장하기 위한 프레임워크

[6]본 절에서 CXL 3.0은 3.1, 3.2 등 이후 모든 CXL 3.x 버전을 포함하여 지칭한다.

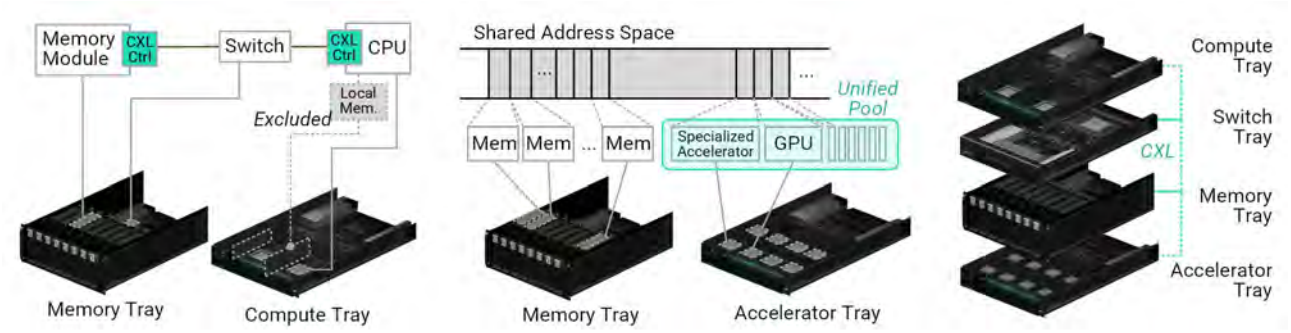
등의 복잡한 소프트웨어의 개입 없이 직접적이며 일관성 있게 공유할 수 있다. 또한, 가속기 내의 로컬 HBM을 단일한 일관된 메모리 풀로 통합할 수 있어 시스템의 메모리 자원 활용도를 더욱 높일 수 있다. 이와 같은 진정한 메모리 공유 방식이 주는 성능적 이점과 실질적 의미는 이후 5.2절에서 자세히 다룬다.

다단계 스위치 토폴로지를 명시적으로 지원하여 인터커넥트 패브릭 구성이 가능하며, 접근 가능한 CXL 장치 수를 크게 확장했다. 또한 다중 호스트 간 소프트웨어 개입 없이 초고속 데이터 공유가 가능하다.

CXL 3.0은 앞서 언급된 상당한 기술적 개선에도 불구하고 CXL 2.0과 하위 호환성(Backward Compatibility)을 유지하여 시스템 확장성을 최대화한다. 하지만 실제 시스템 구현 및 배포 시에는 CPU, 스위치, 엔드포인트 장치에서 특정한 하드웨어 수정이 요구된다. 특히, PBR 메커니즘은 주로 스위치 내부에서 구현되나, 엔드포인트에서도 그에 상응하는 하드웨어 변화가 필수적이다. 예를 들어, 엔드포인트는 기존의 68바이트 "플릿(Flit)" 대신 256바이트 플릿을 처리할 수 있도록 설계되어야 한다. 또한 진정한 다중 호스트 메모리 공유와 캐시 일관성을 실현하기 위해, 메모리 확장 장치는 "백-인밸리데이션(Back-Invalidation)"과 같은 고급 일관성 유지 메커니즘을 반드시 구현해야 하며, 모든 공유 자원 간의 데이터 일관성과 무결성을 엄격히 보장해야 한다. 이 외에도 P2P 직접 통신과 같은 CXL 3.0의 새로운 기능을 통해 가속기는 호스트의 중재 없이 동일 도메인의 다른 가속기나 엔드포인트 메모리에 직접 접근할 수 있어야하며, 이러한 기능을 제공하는 CXL 3.0 컨트롤러를 기존 가속 하드웨어에 적절히 통합하여 설계하고 구현해야 한다. CXL 3.0의 이러한 고급 기능 구현은 PBR 기반의 패브릭 아키텍처가 제공하는 확장성과 유연성을 극대화하고, 전체 시스템의 통신 지연 및 오버헤드를 최소화할 수 있도록 지원하기 때문에 대규모 AI 인프라의 효율적 확장을 지원하기 위해 필수적으로 적용될 필요가 있다.

4.3. CXL 기반의 AI 데이터센터를 위한 모듈형 트레이 및 랙 아키텍처

CXL 기술 기반의 모듈형 트레이 설계. 본 절에서는 CXL 3.0에서 도입된 컴포저블 아키텍처와 다중 호스트 메모리 공유 기능이 실제 현대 AI 데이터센터에서 어떻게 "모듈형(Modular)" 및 "트레이 기반(Tray-Based) 시스템 설계"를 가능하게 하는지 설명한다. 기존 방식에서는 가속기, 메모리, CPU가 물리적으로 밀접하게 결합되어 있어 자원을 독립적으로 확장하거나 유연하게 재구성하기 어려웠다. 반면, CXL 기반의 트레이 방식 시스템은 엔드포인트가 공간적으로 분리된 배치 구성을 할 수 있기 때문에 가속기, CPU, 메모리와 같은 자원을 각기 별도의 트레이로 구성하여 독립적인 확장과 동적 재구성을 쉽게 만든다. 이를 통해 자원 관리가 단순해지고, 변화하는 AI 워크로드의 요구에 신속하게 대응할 수 있도록 구성할 수 있다.



(a) 분리형 메모리 트레이. **(b)** 분리형 가속기 자원 풀. **(c)** 트레이-간 연결.

|그림 26| CXL의 캐시 일관한 공유 특성을 활용한 트레이 기반 모듈형 시스템.

그림 26a와 같이, "메모리 트레이"는 전용 CXL 컨트롤러나 CXL 스위치를 통해 DRAM 모듈만을 모아서 구성되며 이러한 트레이들을 다시 모아 하나의 큰 메모리 풀을 만들 수 있다. 각각의 메모리 트레이는 특정 CPU나 가속기 노드에 고정되지 않고, 여러 가속기나 CPU 트레이 간에 유연하게 할당되고 공유될 수 있다. 예를 들어, 대규모 트랜스포머 모델의 학습 단계에서 더 많은 메모리가 필요하면 CPU 설정을 변경하지 않고 추가 메모리 트레이를 가속기 트레이에 연결하여 메모리를 늘릴 수 있다.

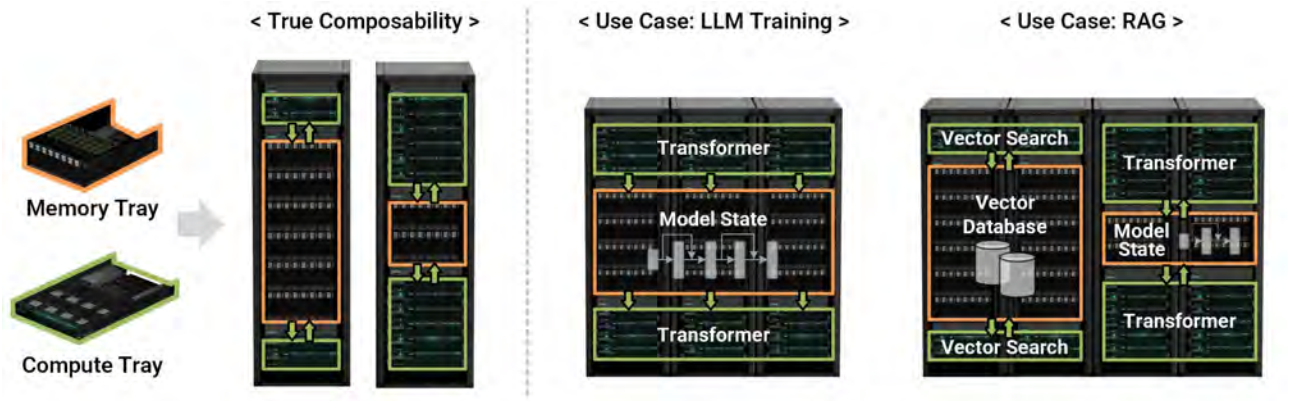
자원을 독립적으로 분리하려는 시도는 이전에도 있었지만 [251–254], 기존 시스템에서는 실제로 자원을 완전히 분리하는 것이 어려웠다. 이전에는 CPU가 메모리 컨트롤러와 캐시 일관성을 직접 관리해야 했기 때문에 메모리와 가속기가 CPU에 의존적이었다. 하지만 CXL은 메모리 컨트롤러를 CPU 외부로 분리하고, 모든 구성 요소 간에 표준화된 캐시 일관성 통신을 지원함으로써 이런 한계를 해결한다. 그 결과, 가속기가 CPU의 개입 없이 직접 메모리 자원을 접근하는 데이터를 효율적으로 공유할 수 있다.

이 모듈형 시스템에서, "전용 가속기 트레이"는 여러 GPU나 NPU, 그리고 도메인 특화 가속기를 고속 CXL 인터페이스로 연결하여 구성된다. 그림 26b에 나타난 바와 같이, 표준화된 CXL 인터페이스를 사용하면 가속기들이 CPU나 메모리 장치와 미리 할당되어 고정된 연결 없이도 하나의 통합된 자원 풀로 묶일 수 있다. 또한 각 가속기의 로컬 메모리를 하나의 공유 메모리 공간으로 통합하여 중복된 데이터 이동을 줄이고 성능을 높일 수 있다. 예를 들어 KV 캐시, 중간 활성화와 같은 자주 접근하는 데이터 구조와 동기화 이벤트들은 효율적으로 관리하여, 이로 인해 발생한 다양한 오버헤드들을 원천적으로 제거할 수 있다.

한편, "컴퓨트 트레이"는 CPU와 필요에 따라 NIC만을 포함하며, 로컬 메모리는 의도적으로 제외하여 자원의 독립성과 다른 자원과의 조합을 통한 확장성을 명확히 제공한다. 여기에 추가로, 전용 CXL 스위치 트레이는 독립적으로 확장 가능한 컴퓨트, 메모리, 가속기 트레이 간의 실시간 자원 할당과 동적 구성을 조율한다.

이렇게 각 트레이가 특정 자원 유형만을 전담하도록 구성되면, CXL 기반 자원 분리의 이점을 최대한 활용하며 워크로드 변화에 따라 컴퓨트, 메모리, 가속기 등 각 자원은 독립적으로 확장 및 재구성할 수 있다. 이러한 CXL 기반의 모듈형 아키텍처는 자원 운영 효율성을 높이고 변화하는 AI 워크로드에 빠르게 대응할 수 있는 유연성을 제공하며, 이후 설명할 랙 수준의 컴포저블 아키텍처의 기반이 될 수 있다.

CXL 기반의 컴포저블 랙 아키텍처. CXL을 이용한 모듈형 트레이 기반 아키텍처가 랙 수준으로 확장될 때에는



|그림 27| CXL 컴포저블 랙 아키텍처 예시.

기능적으로 명확하고 자원의 효율성을 극대화할 수 있도록 구성된다. 각각의 랙은 네트워크, 가속기, CPU, 컴포저블 메모리 확장 장치, 스토리지(Storage)와 같은 자원별로 전용 트레이들로 나뉘어 구성될 수 있다. 이를 통해 특정 응용 프로그램이 필요로 하는 자원을 제공하기 위해, 특화된 랙들로 구성하고 연결하여, 컴포저블한 열을 형성하거나, 한 랙 내부에서 다양한 유형의 트레이를 필요에 따라 유연하게 결합할 수 있다.

그림 26c는 CXL 기반 컴포저블 랙 아키텍처에서 모듈형 트레이들이 어떻게 연결되는지를 보여주는 대표적인 예시이다. 여기서는 메모리 트레이가 랙의 중심에서 고속 CXL 스위치를 통해 관리되는 대규모 메모리 풀을 형성하며, 주변의 가속기 트레이와 컴퓨트 트레이가 이 메모리 풀과 연결되어 있다. 이 방식으로 CPU의 직접적인 개입 없이도 메모리 자원을 효율적이고 신속하게 공유할 수 있어 중복된 데이터 이동을 최소화할 수 있다. 이는 특히 트랜스포머 모델의 추론 작업이나 KV 캐싱처럼 자주 쓰이는 데이터를 반복적으로 활용하는 워크로드에서 성능 개선 효과가 크다.

또 다른 구성 사례로 다양한 AI 워크로드에 적합한 컴포저블 랙 아키텍처의 예를 그림 27에서 보여준다. 여기서는 가속기 트레이와 메모리 트레이가 완전히 분리되어 있어, 각 트레이를 필요에 따라 독립적으로 확장하거나 유연하게 자원을 재구성할 수 있다. 이 구성에서 또한 CXL을 통한 고속 캐시 일관성 통신을 활용하여 가속기 간의 효율적인 데이터 공유가 가능하고, 이를 통해 전체적인 연산 성능과 자원 활용도를 높일 수 있다. 특히 대규모 LLM 학습과 같이 메모리나 연산 요구가 자주 변화하는 워크로드에서 이런 방식이 매우 효과적이다.

이러한 모듈형 아키텍처의 또 다른 중요한 장점은 랙 내부 네트워킹 및 랙 간 네트워킹을 최적화할 수 있다는 것이다. 구체적으로, 기존의 데이터센터는 주로 각 랙 상단에 위치한 ToR 네트워크 스위치를 통해 랙 간 통신을 처리했기 때문에, 가속기 간의 고속 통신이 어렵고 성능 저하가 발생하였다. 예를 들어, 고속 링크인 UALink나 NVLink가 없는 경우에는 랙 내부의 통신조차 장거리 네트워크를 통해 처리해야 했고, 이로 인해 성능이 크게 떨어졌다. 또한 UALink나 NVLink가 지원되는 경우에도, 랙 내부의 다른 장비는 여전히 저속 네트워크를 써야 했기 때문에 병목 현상이 자주 발생했다.

반면 CXL 기반의 모듈형 아키텍처는 이러한 문제를 해결하기 위해 전통적인 ToR 스위치 대신 중앙에 위치한 전용 CXL 스위치 트레이를 사용하며, 이를 랙 중앙부(Middle-of-Rack, MoR)에 배치한다. 더 많은 대역폭이 필요할 경우 추가적인 CXL 스위치 트레이를 쉽게 추가하여 연결성을 강화할 수 있다. 이렇게 되면 랙과 열 내부가 통합된 고속 인터커넥트만으로 연결되는 스케일업(Scale-up) 구성이 가능해진다. 반대로 열 간 통신은 기존의 이더넷이나 인피니밴드와 같은 네트워크 장치를 이용하여 별도의 전용 네트워크 랙에서 처리한다. 이러한 구조는 네트워크의

|표 2| CXL 적용 전후 아키텍처 성능 비교.

현대 AI 인프라의 핵심 성능 지표	기존 아키텍처	CXL 기반 트레이 아키텍처
확장성	자원 확장이 제한적인 노드 또는 랙 수준 스케일업	연산, 인터커넥트, 메모리 자원의 유연한 확장을 가능하게 하는 열 수준 스케일업
지연	높은 지연, RDMA로 인한 프로토콜 오버헤드 및 소프트웨어 개입 (1 µs 이상)	낮은 지연, 하드웨어 기반 프로토콜 관리 및 캐시 일관성 유지 (100–250 ns)
메모리 용량	낮고 고정됨, 밀접하게 통합된 CPU와 GPU 아키텍처 (GPU당 192–288 GB)	막대하고 유연함, 동적으로 구성 가능한 메모리 풀 (노드당 수십 TB 이상)
메모리 대역폭	낮은 효율 (외부 메모리 접근 시의 메모리 복사)	높은 효율 (일관성이 보장되는 풀링 메모리를 통한 트래픽 감소)
연산 자원의 유연성	낮은 유연성 (고정적이거나 큰 단위의 자원 할당)	높은 유연성 (세밀한 단위의 동적 자원 할당)

혼잡을 줄이고 지연을 최소화하여 전체적인 성능을 향상시키며, AI 워크로드의 요구를 효율적으로 만족시킬 수 있다.

운영 측면에서도 이 트레이 기반 설계는 유지보수 및 업그레이드를 훨씬 쉽게 만들어 준다. 메모리, 가속기, CPU, 네트워크 장비가 물리적·논리적으로 명확하게 분리되어 있기 때문에, 특정 자원을 교체하거나 확장할 때 전체 시스템을 중단하지 않고도 개별 자원만 빠르게 교체하거나 추가할 수 있다. 이를 통해 시스템의 가동 중단 시간을 최소화하고, 신기술 도입을 용이하게 하며, 인프라가 빠르게 변화하는 워크로드 요구에 민첩하게 대응할 수 있도록 지원하여 장기적으로 비용 효율성을 높일 수 있다.

CXL 아키텍처를 통한 다양한 성능 지표 동시 충족 방안. 표 2에서는 CXL을 이용한 트레이 방식 아키텍처와 랙 구성이 현대 AI 인프라에서 요구하는 중요한 성능 요소들을 어떻게 효과적으로 충족하는지 분석한다. 이 모듈형 아키텍처는 유연한 자원 배치와 확장을 가능하게 하여 연산의 유연성을 높이고, 메모리 용량과 대역폭을 최적화하며, 지연을 줄이고 시스템 전체의 확장성을 개선할 수 있다.

다음은 CXL 기반 아키텍처가 4.1절에서 소개된 각각의 성능 지표를 만족시키는 방식을 더 쉽게 설명한 것이다.

- **확장성(Scalability):** CXL은 여러 단계의 스위치를 서로 연결하여 확장성 있는 구조를 만든다. 이러한 구조는 자원을 추가할 때 병목 현상이 생기지 않도록 하며, 수천 개의 가속기와 메모리를 하나의 통합된 자원 풀로 묶을 수 있다. 이를 통해 시스템은 점점 커지는 AI 모델 크기와 병렬 워크로드를 효과적으로 지원하며, 추가적인 하드웨어 변경 없이도 다양한 AI 요구에 지속적으로 대응할 수 있다.
- **지연(Latency):** CXL은 메모리와 가속기 간에 직접적이고 캐시 일관성을 유지하는 통신 경로를 제공하여 지연 문제를 줄인다. 기존의 네트워크 기반 방식(RDMA 등)은 소프트웨어 처리가 많아 지연이 컸지만, CXL은 하드웨어 기반으로 직접 메모리에 접근하기 때문에 지연을 크게 줄일 수 있다. 이러한 특징은 LLM 추론이나 디코딩처럼 빠른 응답이 중요한 작업에서 특히 유리하다. 데이터 접근에 있어 소프트웨어 개입을 없애고 각 연산장치의 온칩(On-Chip) 캐시를 사용하기 때문에 시스템 전반의 성능과 효율성을 높일 수 있다.
- **메모리 용량 및 대역폭(Memory capacity and bandwidth):** CXL 기반 메모리 트레이는 여러 개의 메모리 모듈을 한데 묶어, 높은 용량과 대역폭을 제공한다. 각 메모리 모듈은 전용 CXL 컨트롤러나 스위치를 통해 연결 및 확장되며, 가속기가 이 메모리 풀에 직접 접근하여 메모리 용량 부족이나 데이터 이동 문제를 해소할 수 있다. 특히 CXL의 캐시 일관성 기능(CXL.cache)을 활용하면 메모리를 공유하는 가속기 간 캐시 상태가

동일하게 되어 불필요한 데이터 전송과 메모리 접근을 줄일 수 있다. 이러한 특성을 이용하면 RAG나 KV 캐싱처럼 데이터 접근이 많은 작업에서 성능이 크게 향상된다.

- **연산 자원의 유연성(Computational flexibility):** CXL 기반 트레이 아키텍처는 다양한 자원(가속기, 메모리, CPU)을 독립적으로 조합하고 변경할 수 있게 한다. 이를 통해 워크로드의 특성에 따라 필요한 자원을 비용 효율적으로 확장하거나 정밀하게 구성할 수 있다. 예를 들어, GPU 트레이는 연산이 많은 학습 단계나 프리필 단계에서 추가로 확장될 수 있고, 빠른 응답이 필요한 추론 단계에서는 더 낮은 지연을 제공하도록 동적으로 재구성할 수 있다. 이렇게 유연한 자원 관리는 연산 능력과 메모리 용량 등을 최적으로 활용하여 전체 시스템 효율성을 높일 수 있다.

CXL 기반 인프라는 동적이고 빠르게 변화하는 AI 워크로드를 효율적으로 수용하며 다수 성능지표를 동시 만족할 수 있도록 도울 수 있다. 또한 CPU 개입 없이 다양한 가속기가 공유 메모리를 직접 접근할 수 있도록하여 성능과 비용 효율성을 극대화 할 수 있다.

이러한 아키텍처적 특징을 통합적으로 활용하면, CXL 기반 인프라는 빠르게 변화하는 AI 워크로드의 요구사항을 효율적으로 수용하고 동적으로 대응할 수 있다. 특히 확장 가능한 스케일업 도메인은 자원 간의 통신 오버헤드를 크게 줄여 전체 성능을 향상시키며, 불필요한 추가 스위치를 최소화하여 TCO도 낮출 수 있다. 또한 가속기는 CPU의 개입 없이 공유된 메모리를 직접 접근할 수 있어 데이터 전송 부담이 줄어들고, 결과적으로 연산 성능 및 비용 효율성이 향상된다. 따라서, CXL 기반의 모듈형 아키텍처는 대규모 AI 워크로드가 요구하는 다양한 성능 지표를 효과적으로 지원하는 유연하고 강력한 솔루션이 될 수 있다.

5. 컴포저블 CXL 아키텍처: 통합 전략과 실증 평가

이제까지 컴포저블 CXL 아키텍처가 현대 AI 인프라에서 발생하는 확장성 및 성능 문제를 어떻게 해결할 수 있는지 살펴보았다. 특히 메모리와 가속기 자원을 동적으로 분리하고 통합하는 방식을 통해, 기존의 RDMA 방식보다 높은 유연성과 적응성을 제공한다는 점을 강조했다. 그러나 지금까지 논의한 구조는 주로 CXL 기반의 모듈형 트레이 상위 수준 설계에 초점이 맞춰져 있었다. 실제 구현을 위해서는 개별 트레이를 구체적으로 어떻게 구성하고, 트레이 내부에서 GPU와 가속기 간 연결을 어떻게 효율적으로 구성해야 하는지에 대한 심도 있는 논의가 필요하다.

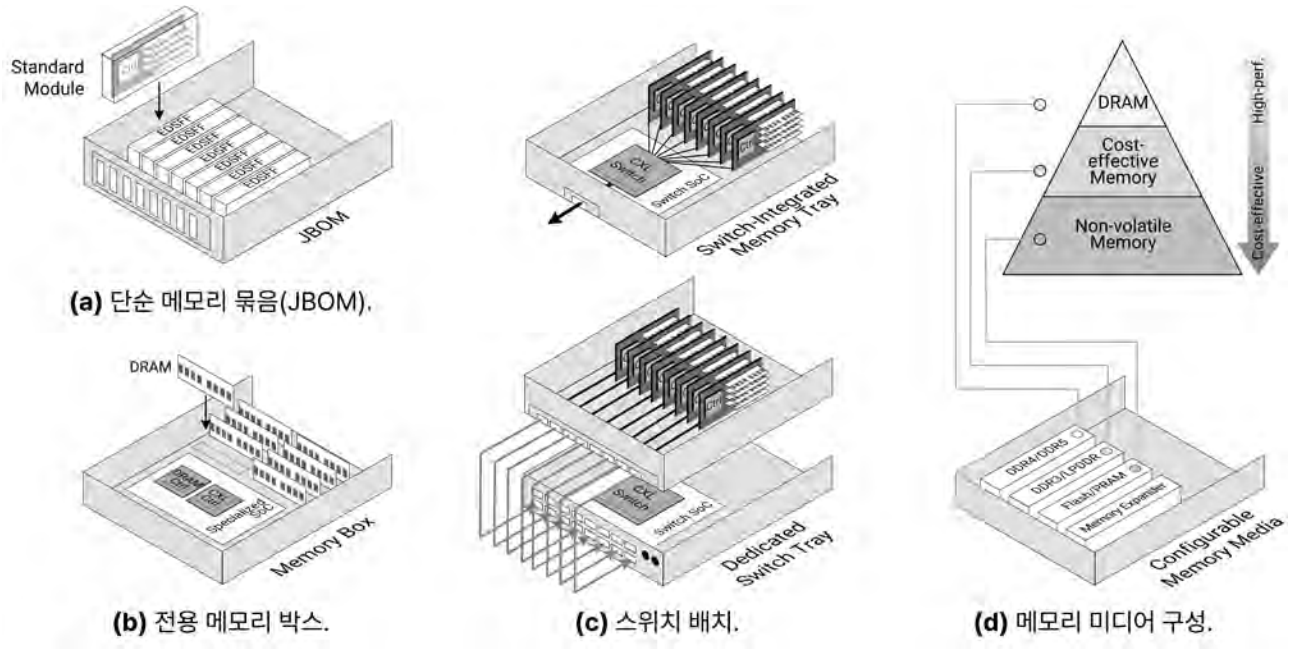
본 절에서는 CXL 기반 컴포저블 인프라의 구체적인 구현 방법에 집중한다. 이를 위해 구체적인 통합 전략을 제시하고, 다양한 실제 워크로드를 사용한 평가와 사용 사례를 살펴보며 그 성능과 효율성을 입증하고자 한다.

5.1. 컴포저블 CXL 데이터센터의 메모리 및 가속기 관리

이 하위 절에서는 다음의 주요 요소들을 중심으로 논의한다. 먼저 전용 메모리 풀을 구성하기 위한 다양한 아키텍처적 요구사항과 구현방향을 알아보고, 이어서 대규모 AI 인프라내에서 가속기의 효율적인 할당과 연결 전략을 설명한다. 마지막으로 메모리와 가속기 자원을 일관성 있게 관리할 수 있는 통합 소프트웨어 프레임워크의 개발 방향을 다룬다.

전용 메모리 풀의 구현과 관리. 전용 메모리 풀을 구현하고 관리할 때는 다음의 핵심적인 요소들을 고려해야 한다. 첫 번째는 메모리 트레이 자체의 하드웨어 구조와 구체적인 구현 방식을 결정하는 것이다. 두 번째는 비용 효율적으로 CXL 스위치를 배치하고, 이 스위치가 컴포저블 시스템 내에서 어떤 역할을 수행할지 정의하는 것이다. 마지막으로는 메모리 풀에 적합한 후단(Backend) 메모리 미디어를 신중히 평가하여 선택해야 한다.

먼저 메모리 트레이의 하드웨어 구성 방식부터 살펴보자. 그림 28(a)에 나온 것처럼, 메모리 트레이는 "단순 메모리 묶음(JBOM, Just Bunch of Memory)" 형태 또는 전용 메모리 박스(Dedicated Memory Box) 형태로 구현될 수

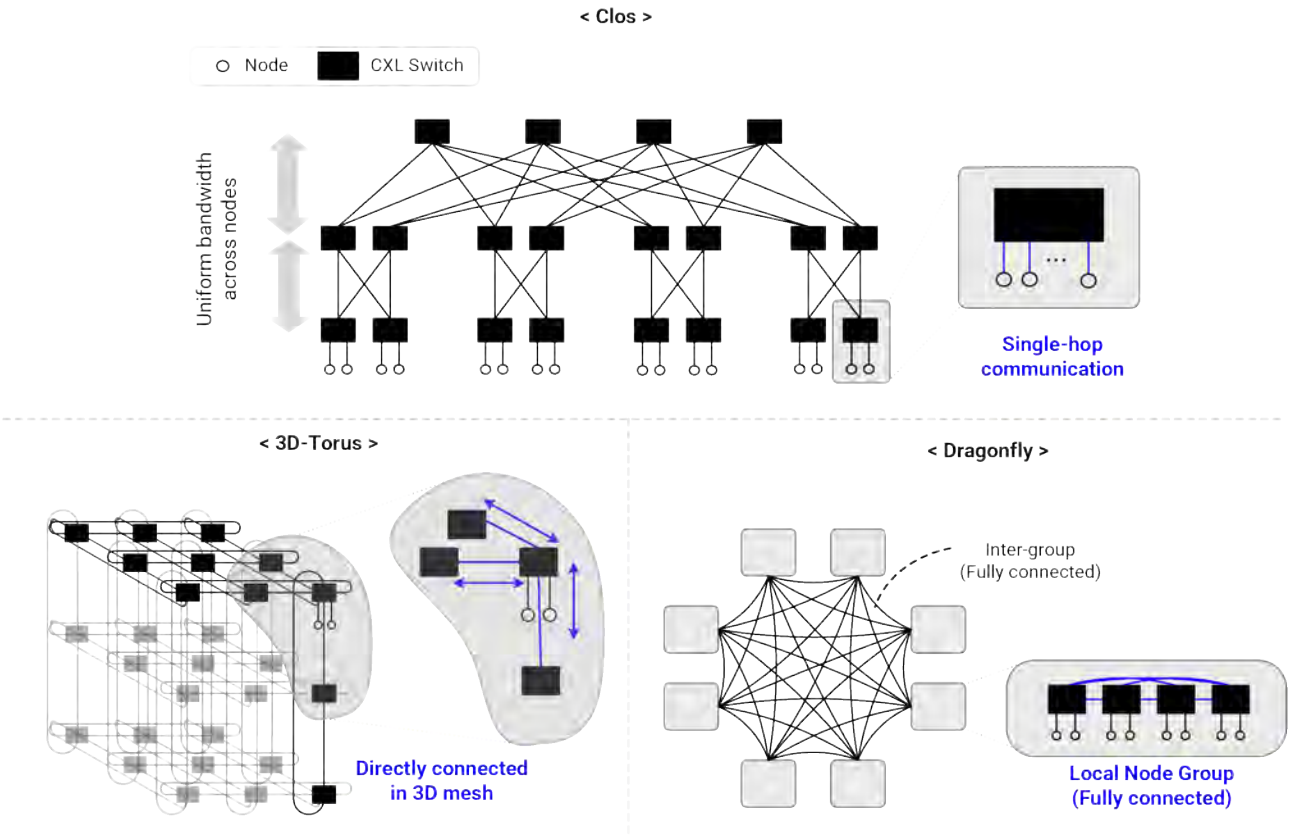


|그림 28| 트레이 기반 전용 메모리 풀 구현 및 관리 전략.

있다. JBOM 방식에서는 메모리 확장 장치를 데이터센터 표준인 EDSFF(Enterprise and Datacenter Storage Form Factor [335–338]) 모듈 형태로 배열하여 사용한다. 이는 많은 제조사들이 사용하는 표준 방식이지만, 제조사 간 성능 차이와 운영 관리의 복잡성으로 인해 비용과 관리 부담이 증가할 수 있다. 특히, 메모리 미디어 교체 시 더 긴 수명을 가진 CXL 및 메모리 컨트롤러도 함께 교체해야 하기 때문에 TCO가 상승할 수 있다.

반면, 전용 메모리 박스 형태는 별도의 시스템 온 칩(SoC, System-on-Chip)에 여러 개의 DRAM과 CXL 컨트롤러를 통합하여 구성한다. 그림 28(b)에 나타난 것처럼, 이 방식은 메모리 컨트롤러를 메모리 미디어로부터 분리하여, 컨트롤러로 하여금 저수준 DRAM칩이나 듀얼 인라인 메모리 모듈(DIMM, Dual Inline Memory Module [339–341])을 직접 활용할 수 있게 한다. 이러한 분리 방식은 데이터센터 입장에서 유지보수와 운영 비용을 현저히 줄이고, 오래된 DIMM이나 기존 DDR 메모리 모듈을 재사용할 수 있어 추가적인 비용 절감 효과를 얻을 수 있다. 또한 사용자는 포트 수, 대역폭, 메모리 유형 등을 워크로드에 맞게 유연하게 구성할 수 있으며 DDR3 [342–345], DDR4 [346–348], LPDDR [349–351] 등의 다양한 메모리를 필요에 따라 도입하여 성능과 비용 간 균형을 맞출 수도 있다. 다만, 이 방식은 데이터 무결성 관리나 이기종 하드웨어 간의 복잡한 설계 관리와 같은 추가적인 복잡성이 발생할 수 있다.

다음으로, 메모리 확장 장치와 SoC 컨트롤러를 효과적으로 연결할 최적의 CXL 스위치 배치 방안을 결정해야 한다. 그림 28(c)에 나타난 것처럼, 메모리 트레이 내부에 직접 CXL 스위치를 통합하면 여러 장치 간의 호환성 문제나 성능 차이를 쉽게 관리할 수 있다. 각 트레이를 독립된 장치로 취급함으로써 앞서 언급된 JBOM과 같이 내부 하드웨어의 변화를 외부에 노출하지 않고, 일관된 성능과 표준화된 인터페이스를 제공할 수 있다. 이 방식은 메모리 트레이를 더욱 긴밀하게 통합하지만, 트레이당 비용이 높고 하드웨어 유연성이 떨어질 수 있다. 반대로, 스위치를 외부의 전용 트레이나 랙 중앙(MoR) 또는 상단(ToR)에 집중적으로 배치하면 메모리 트레이를 단순화하고 재사용하여 비용을 낮추고, 다양한 데이터센터 환경에서의 유연한 배치가 가능해진다.

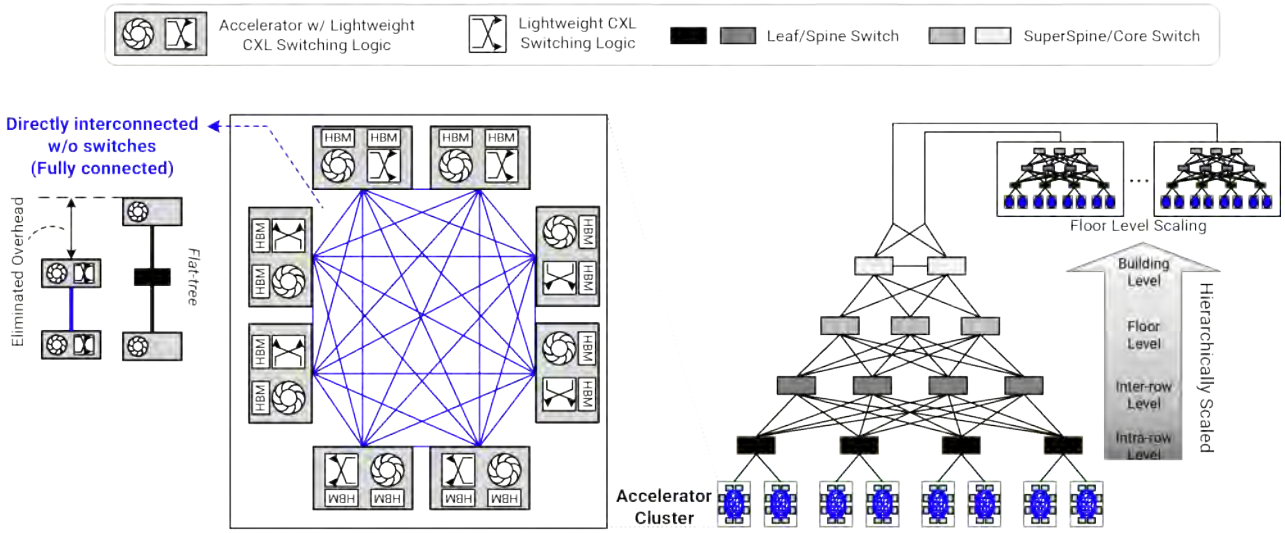


|그림 29| 토폴로지 비교: 클로스(Clos), 3D-토러스(3D-Torus), 드래곤플라이(Dragonfly).

마지막으로, 다양한 메모리 미디어를 트레이 내부에서 계층적으로 구성하여 전체 효율을 높이도록 구현할 수 있다(그림 28(d)). 최신 DRAM(DDR4/DDR5)은 비용이 높고 유연성이 제한적이기 때문에, 비용 효율이 좋고 전력 소모가 낮은 모바일 메모리(LPDDR)이나 구형 DRAM(DDR3 등)와 같은 메모리를 선택적으로 사용할 수 있다. 추가적으로, 고성능의 HBM 모듈을 트레이 내부 중간 계층(Intermediate Buffer Layer)으로 활용하여 LPDDR이나 DDR3와 같은 메모리 모듈의 성능을 높이고, 메모리 확장 장치 간 성능 차이를 보완하여 지연을 줄일 수 있다. 또한 플래시 메모리나 PRAM과 같은 비휘발성 메모리를 도입하면 데이터 지속성을 제공하면서 성능과 비용 측면에서 추가적인 최적화 효과를 얻을 수 있다.

가속기 자원 관리: 토폴로지와 상호 연결 전략. 가속기의 자원을 효율적으로 관리하려면 단순히 서로 연결하는 것 이상으로 여러 요소를 신중하게 고려해야 한다. 특히, 토폴로지 설계는 가속기 제조사의 다양한 특성과 워크로드별 통신 요구를 충분히 반영하여 결정되어야 한다. 본 절에서는 텐서 병렬화(Tensor Parallelism)를 활용하며, 데이터의 지역성(Data Locality)이 높고 인접 가속기 간에 데이터 교환이 집중되는 LLM 워크로드를 중심으로, 가속기 자원 관리의 주요 사항을 논의한다. 임의의 일반적인 트래픽을 처리하는 범용 AI 데이터센터를 위한 하이브리드 연결 방식은 이후 6절에서 별도로 다룬다.

1차원적인 연결 구성만을 제공하는 UALink나 NVLink와 같은 기존의 가속기 전용 인터커넥트 기술과 비교할 때, CXL은 다양한 형태의 연결 구성을 지원하여 데이터센터에서 다양한 자원을 유연하게 연결하도록 해준다. 그림 29에



(a) 가속기-간 직접 연결된 가속기 클러스터. **(b)** 계층적으로 확장된 가속기 클러스터.

|그림 30| 스위칭 로직을 포함하는 가속기 중심 통합 전략.

서는 현대 데이터센터에서 흔히 쓰이는 클로스 [256, 352, 353], 3D-토러스 [354–357], 드래곤플라이 [358–360]의 주요 특성을 비교하였다. 먼저 클로스 토폴로지는 다단계의 스위치를 통해 모든 노드 간 균일한 대역폭을 제공하며 확장성이 매우 뛰어난 토폴로지를 제공하지만, 인터커넥트 패브릭 구성의 구축 비용과 복잡성이 높다는 단점이 있다. 3D-토러스 토폴로지는 3차원 메쉬 형태로 노드를 직접 연결하여 근거리 통신을 효율적으로 처리하지만, 먼 거리의 데이터 교환에서는 병목이 발생할 수 있다. 드래곤플라이 토폴로지는 로컬 그룹 간 완전 연결과 그룹 간의 간접 연결을 조합하여 성능과 비용의 균형을 유지하지만, 특정 통신 패턴에서는 효율성이 떨어질 가능성이 있다.

LLM 워크로드의 특징상(예: 인접 가속기 간의 집중적인 데이터 교환), 3D-토러스나 드래곤플라이가 적합해 보일 수 있지만, 가속기의 수가 많아질수록 요구되는 스위치의 개수가 급격히 증가하여 대규모 데이터센터 구축에 현실적으로 사용이 어렵다. 따라서 캐시 일관성을 유지하면서 가속기를 연결하는 "단일 홉"의 평면형 클로스 토폴로지를 사용하는 것이 더 실용적이다. 이러한 설계는 NVLink나 UALink와 같은 기존 가속기 전용 인터커넥트와 유사한 전략으로, 합리적인 비용으로 효율적인 로컬 통신을 가능하게 한다. 다만 현재의 CXL 사양은 캐시 일관성 연결이 가능한 가속기의 수를 최대 256개로 제한하고 있어, 더 큰 규모의 연결에는 다른 전략이나 추가적인 스위치가 필요할 수 있다.

더욱 적극적인 통합 방안이 그림 30a에 제시되어 있다. 이 방법은 중간에 별도의 CXL 스위치를 두지 않고 가속기들이 직접 완전 연결 토폴로지를 형성하는 방식이다. 각 가속기가 내부에 경량화된 CXL 스위칭 로직을 직접 통합하여, 추가적인 외부 연결 장비가 필요 없게 된다. 완전 연결 방식의 아키텍처는 인접한 가속기 간 데이터 전송을 최적화하여, LLM처럼 로컬 통신이 빈번한 워크로드에서 특히 높은 성능을 보인다. 이렇게 구성된 가속기 클러스터는 그림 30b에 나타난 것처럼 랙이나 플로어 등으로 외부의 CXL 스위치를 통해 계층적으로 확장할 수 있다. 또한 가속기 내에 위치한 로컬 HBM을 적극적으로 활용하면 내부 통신 효율성을 더욱 높일 수 있다. 다만, 이 완전 연결 방식은 가속기와 Type 1 및 Type 2 장치에 들어가는 CXL 컨트롤러 설계를 복잡하게 만들 수 있으며, 클러스터 간 외부 통신 시 성능 불균형 문제가 발생할 가능성도 있다. 따라서 실제 대규모로 구축할 때는 성능 평가와 설계를 신중히 진행해야 한다.

컴포저블 자원을 위한 통합 관리 프레임워크. AI 데이터센터에서 컴포저블 CXL 인프라를 실제로 구현할 때는 소프트웨어와 관련된 여러 설계 요소를 신중하게 고려해야 할 수 있다. 기존의 정적인 메모리 할당 방식(Static Memory Allocation)은 추천 시스템의 임베딩 테이블이나 트랜스포머 모델의 어텐션 캐시와 같이 빈번히 접근하는 데이터에서 지나친 데이터 이동과 높은 지연을 유발할 수 있다. 이러한 문제를 극복하려면 고급 소프트웨어 프레임워크를 도입하여 동적이고 효율적인 메모리 관리를 지원해야 한다. 예를 들어, 접근 빈도를 예측하여 데이터를 미리 적합한 메모리 위치로 이동시키는 예측 기반 메모리 배치(Predictive Memory Placement), 미리 캐시를 준비하는 선제적 캐시 워밍(Proactive Cache Warming), 그리고 실시간 접근 패턴에 따라 캐시에서 데이터를 유연하게 제거하는 적응형 퇴출 정책(Adaptive Eviction Policy) 등을 도입할 수 있다. 특히, 텐서 병렬 방식을 사용하는 LLM의 추론 과정에서 자주 접근하는 어텐션 캐시를 가속기 가까운 곳에 동적으로 배치하면 성능과 처리량을 크게 향상할 수 있다.

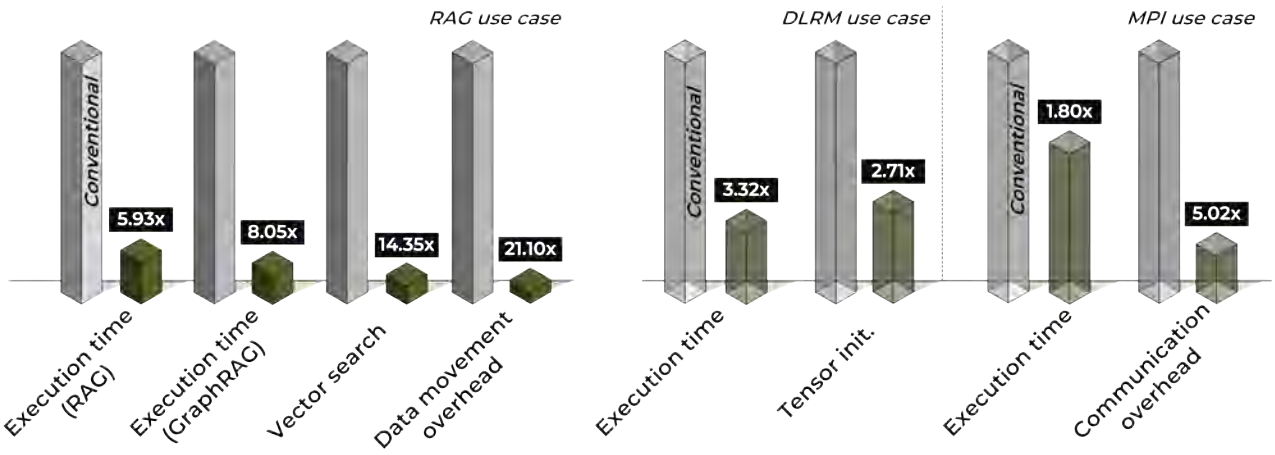
가속기 자원의 효율적인 관리 또한 시스템 전체의 성능을 결정하는 중요한 요소이다. 기존 GPU-CPU 통합 아키텍처는 연산과 메모리를 밀접하게 결합하여 다양한 워크로드 환경에서 가속기의 효율성을 제한하는 경우가 많다. 반면, CXL 기반의 컴포저블 아키텍처는 자원을 독립적으로 확장하고 동적으로 재구성할 수 있으므로 변화하는 워크로드 요구에 빠르게 대응할 수 있다. 이를 효과적으로 달성하려면 워크로드를 실시간으로 모니터링하고, 자원을 미리 예측하여 할당하며, 작업 우선순위에 따라 가속기를 효율적으로 배치하는 소프트웨어 프레임워크가 필요하다. 또한, 빠르게 자원을 재구성할 수 있는 기능과, 하드웨어 자원을 정밀하게 조정하고 가속기 간 통신 지연을 최소화하는 기술도 필수적이다.

메모리와 가속기 자원을 함께 관리하려면 기존의 정적인 방식에서 벗어나 중앙 집중형 모니터링 프레임워크가 필요할 수도 있다. CXL 기반의 컴포저블 시스템에서는 자원의 사용량이 동적으로 변화하므로, 실시간으로 시스템 정보를 수집하고 성능을 분석하여 필요한 조치를 자동으로 수행할 수 있어야 한다. 예를 들어, 실시간으로 원격 측정 데이터를 수집하고(Telemetry), 캐시 일관성 메모리의 효율성을 점검하며, 자원 할당의 지연과, 자원 간 경합 상황을 감지하여 자동으로 문제를 해결하는 기능이 있다면 추가적으로 성능을 향상시킬 수 있을 것이다. 미래에는 강화학습과 같은 고급 기술을 이용하여 자원 배치와 사용을 최적화하고 문제를 예측하여 미리 대응할 수 있는 소프트웨어 프레임워크가 더욱 중요해질 것이다. 이러한 발전된 기능은 AI 워크로드에서 민감한 지연 문제를 해결하여 시스템의 반응성과 효율성을 크게 높일 수 있을 것으로 기대된다.

5.2. AI 워크로드를 위한 CXL 인프라: 실증적 사례 연구

앞서 소개된 CXL의 이론적인 장점들을 바탕으로, 본 절에서는 실제 프로토타입 시스템을 구축하여 CXL 기반 컴포저블 인프라가 실제 환경에서 얼마나 효과적인지를 평가하였다. 특히, RAG, Graph-RAG, 딥러닝 추천 모델(DLRM), MPI 기반 과학 컴퓨팅 등 대표적인 AI 및 HPC 워크로드를 실험하여 기존 아키텍처 대비 CXL의 실질적인 성능 이점을 확인하였다.

실험 결과, 컴포저블 CXL 인프라는 기존 RDMA 시스템에서 빈번히 나타나는 긴 지연, 과도한 데이터 이동 오버헤드, 비효율적인 메모리 활용 문제를 효과적으로 해결하는 것으로 나타났다. 그림 31은 본 절에서 소개될 워크로드별 실험을 통해 기존 아키텍처 대비 CXL 기반 시스템이 얼마나 성능이 향상되는지를 요약한 것이다. 구체적으로, LLM



|그림 31| 성능 향상치 요약: RAG, Graph-RAG, DLRM, 그리고 MPI.

과 결합된 RAG 및 Graph-RAG 기반 AI 검색 워크로드의 실행 시간이 기존 대비 5배 이상 단축되었으며, 데이터 이동 오버헤드는 최대 21.1배까지 감소하였다. 임베딩 중심의 DLRM 워크로드는 추론 실행 시간이 약 3.32배 향상되었고, 텐서 초기화 속도도 2.71배 빨라졌다. 또한, MPI 기반의 HPC 애플리케이션에서도 CXL의 직접 메모리 공유로 실행 시간이 약 1.8배 개선되고, 통신 오버헤드는 최대 5.02배 감소하였다.

이후 하위 절에서는 앞으로 소개될 CXL 프로토타입 실험 환경에서 수행한 다양한 워크로드 평가 시나리오와 그 성능 결과를 구체적으로 설명한다. 또한, 이러한 평가 결과를 바탕으로 CXL 기반 AI 데이터센터 설계를 위한 핵심적인 아키텍처적 시사점과 실질적인 설계 방법 등을 논의해본다.

실험 환경 구성. 컴포저블 CXL 아키텍처를 평가하기 위해 CXL 3.0 표준을 준수하는 실제 시스템 프로토타입을 개발하여 실험 환경을 구축하였다. 그림 32a과 32b는 본 연구에서 사용한 실리콘 검증(Silicon-Proven) CXL 컨트롤러 IP와 실험 환경 설정을 각각 나타낸 것이다. 이 실험 환경에서는 GPU 컴퓨팅 노드와 컴포저블 메모리 확장 장치를 계층적 CXL 스위치 토폴로지로 연결하였다. 메모리 확장 장치와 스위치는 표준 CXL 하드웨어 스택을 사용하며, GPU와 CPU 노드는 자체 개발한 CXL 하드웨어를 루트 포트 및 엔드포인트 컴플렉스에 직접 통합하였다. 현재 상용 CXL 3.0 호환 GPU 및 CPU가 없으므로, 오픈소스 Vortex GPU [361, 362]와 RISC-V CPU [363, 364] 마이크로아키텍처를 사용하여 자체 개발한 CXL 컨트롤러 기능을 포함하도록 수정하였다. 이렇게 수정된 GPU와 CPU 프로토타입은 각각 그림 32c와 32d에서 볼 수 있다.

(a) CXL IPs. **(b)** 실제 시스템 프로토타입. **(c)** CXL GPU. **(d)** CXL CPU.

|그림 32| 실험용 CXL3.0 호환 종단 간 인프라 구조.

구성된 메모리 모듈은 캐시 일관성을 유지하며 동적으로 재구성 가능한 메모리 풀을 형성하고, GPU 컴퓨팅 노드에게는 별도의 NUMA [365–367] 도메인으로 표시되도록 구성되었다. 이러한 컴포저블 설계 덕분에 GPU 노드는 CPU 중심의 메모리 관리나 RDMA 프로토콜 없이도 공유 메모리에 직접 접근할 수 있다. 본 연구에서 사용한 프로토타입은 경량화된 오픈소스 CPU 및 GPU 설계를 기반으로 했지만, 우리가 개발한 CXL 컨트롤러 및 하드웨어 스택 IP는 다양한 서드파티의 GPU, NPU, 메모리 확장 장치 등과도 호환 가능하다. 또한 이 IP는 다양한 캐시 및 시스템 버스 인터페이스를 지원하며 기존 하드웨어 플랫폼에 쉽게 통합될 수 있다.

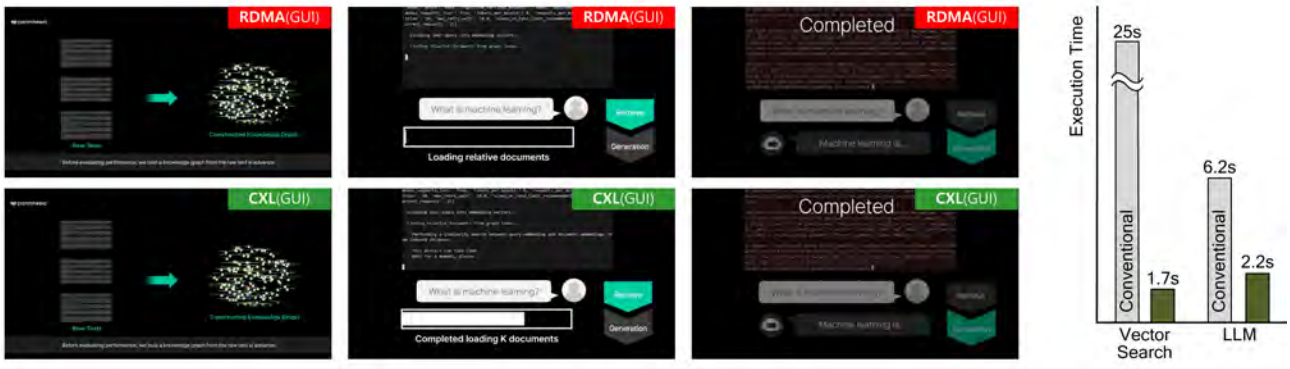
RAG 활용 사례: 인터랙티브 검색 및 추론 가속화. 벡터 매칭과 실시간 추론이 결합된 인터랙티브 검색 작업은 기존 인프라에서 높은 지연과 많은 메모리 사용량 때문에 성능 저하가 자주 발생하는 워크로드이다. 하위 절에서는 컴포저블 CXL 인프라의 실제 효과를 검증하기 위해, 최신 LLM과 결합된 사용자 친화적인 RAG 시나리오를 평가하였다. 그림 33에서 보는 것처럼, 사용자가 자신의 냉장고에 있는 음식 재료의 이미지를 업로드하고 원하는 식사 유형(예: 아침 또는 저녁)을 선택하면, 이에 맞는 요리법을 추천해 주는 시스템이다. 사용자가 올린 이미지는 사전 학습된 비주얼-언어 모델인 CLIP [368–370]을 이용해 임베딩 벡터로 변환되며, 이 벡터는 시스템에 저장된 기존 레시피 임베딩과 벡터 매칭을 통해 비교된다. 기존 RDMA 기반 인프라와 달리, 컴포저블 CXL 아키텍처는 메모리 접근 지연과 소프트웨어 오버헤드를 줄여 벡터 검색 과정에서 성능을 크게 향상시켰다.

검색된 임베딩 벡터는 이어지는 LLM 기반 추론 단계의 입력으로 사용되어, 사용자 맥락에 적합한 요리법을 추천한다. 기존 RDMA 시스템에서는 이 과정이 수백 밀리초에서부터 심지어 수 초대의 지연이 발생하지만, 컴포저블 CXL 인프라는 수십 밀리초 수준으로 응답 속도를 눈에 띄게 개선하였다. 그림 33d에서 나타난 것처럼, 벡터 검색 및 LLM 추론 작업에서 기존 대비 각각 약 14배와 2.78배 빠른 속도를 달성하였다. 이는 사용자 대응 애플리케이션에서 매우 중요한 요소로, 사용자 경험과 만족도를 크게 향상 시킬 수 있다.

Graph-RAG 활용 사례: 지식 그래프 기반 검색 및 추론 가속화. 구조화된 지식 검색을 추론과 결합한 Graph-RAG 워크로드도 메모리 접근 지연 문제로 인해 기존 인프라에서 성능이 저하되는 경우가 많다. 본 연구에서는 Graph-RAG 시나리오를 통해 컴포저블 CXL 인프라의 성능을 평가하였다. 그림 34에서 볼 수 있듯, 평가 과정은 먼저 원본 텍스트 데이터를 RDF 임베딩 [371, 372]이나 그래프 신경망(GNN, Graph Neural Network [373–375])과 같은 기술로 처리하여 효율적인 지식 그래프를 구축한다. 이후 사용자 쿼리는 벡터 형태로 변환되어 구축된 지식 그래프와 빠르게

(a) 초반 (t_0). **(b)** 중반 (t_1). **(c)** 후반 (t_2). **(d)** 단계별 실행 시간.

|그림 33| RAG 활용 사례: 요리법 추천 (데모 동영상: [Link]).



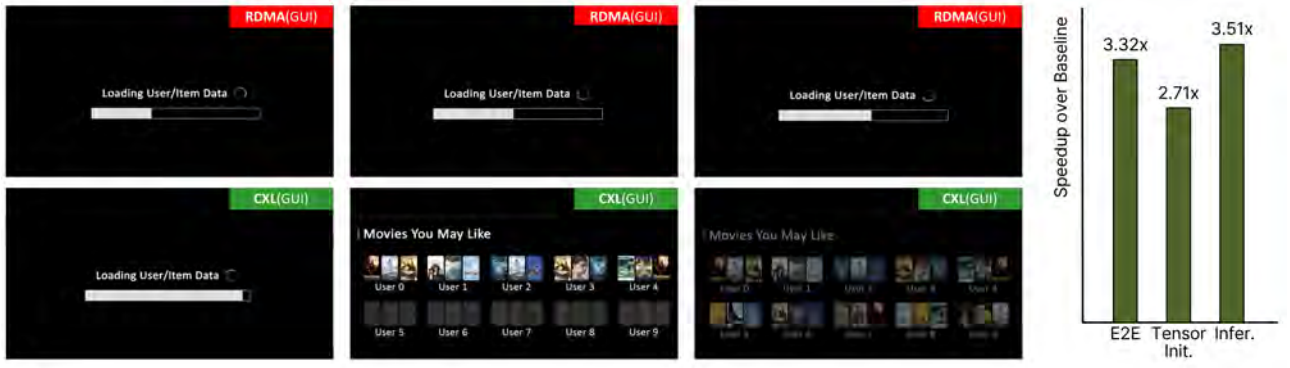
(a) 초반 (t_0). **(b)** 중반 (t_1). **(c)** 후반 (t_2). **(d)** 단계별 실행 시간.

|그림 34| Graph-RAG 활용 사례: 지식 그래프 기반 쿼리 처리.

매칭 [376–381]되도록 하고, 이 결과가 LLM 추론 과정에 입력되어 전체 맥락을 이해하여 보다 정확한 응답을 생성해낸다 [323, 382, 383].

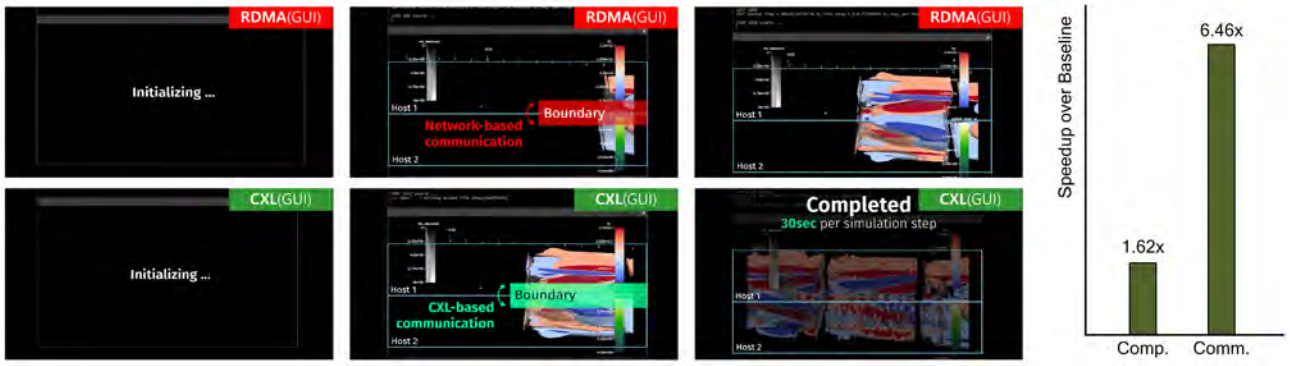
기존의 인피니밴드를 이용한 RDMA 기반 시스템과 비교했을 때, 컴포저블 CXL 아키텍처는 전체 워크플로우 실행 시간을 약 8.05배 단축하였다. 특히 그림 34d에서 볼 수 있듯이, 기존 시스템에서는 수십 초 걸리던 작업이 벡터 검색과 LLM 추론 단계에서 각각 1.7초와 2.2초만에 완료되었다. 이러한 성능 개선은 CXL이 제공하는 캐시 일관성 메모리 풀 덕분이며, CXL 프로토콜과 하드웨어 구조가 Graph-RAG 실행 시 발생하는 데이터 복사 및 소프트웨어 오버헤드를 최소화한 결과이다.

DLRM 활용 사례: 딥러닝 기반 추천 워크로드 가속화. DLRM [384–386] 워크로드는 임베딩 조회의 효율성이 중요하며, 대규모 메모리 요구로 인해 기존 인프라에서 메모리 관련된 성능 문제가 자주 발생한다. 본 하위 절에서는 컴포저블 CXL 인프라를 활용하여, 실제 프로덕션 환경을 대표하는 대규모 임베딩 테이블을 사용한 텐서 초기화 및 추론 단계를 분석 및 비교하였다. 그림 35에 보이듯, 텐서 초기화 단계에서 컴포저블 CXL 인프라는 기존 RDMA



(a) 초반 (t_0). **(b)** 중반 (t_1). **(c)** 후반 (t_2). **(d)** 실행 속도 개선.

|그림 35| DLRM 활용 사례 (데모 동영상: [Link]).

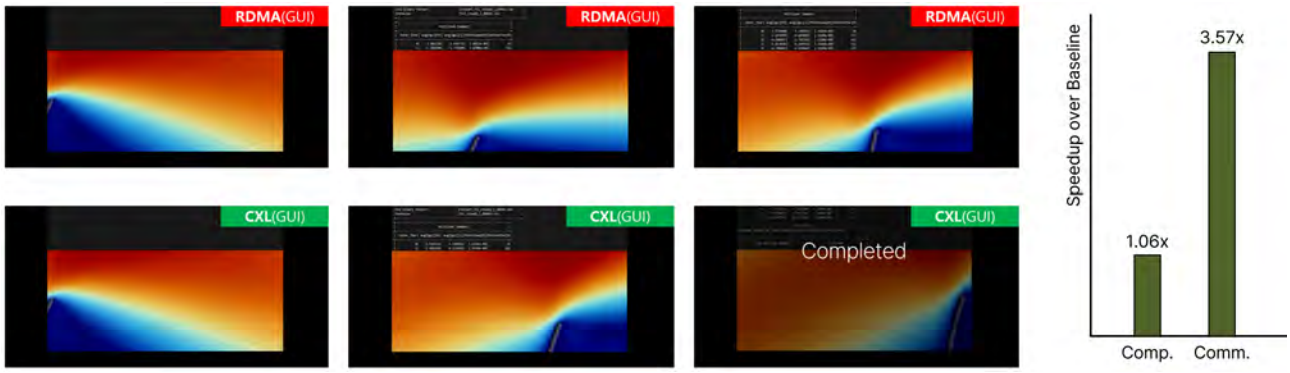


(a) 초반 (t_0). **(b)** 중반 (t_1). **(c)** 후반 (t_2). **(d)** 실행 속도 개선.

|그림 36| MPI 활용 사례: 플라즈마 시뮬레이션 (데모 동영상: [Link]).

기반 시스템 대비 큰 성능 향상을 보였다. 기존 시스템의 소프트웨어 오버헤드와 높은 지연을 컴포저블 인프라가 하드웨어 수준의 직접 메모리 접근 및 캐시 일관성 메모리 풀을 통해 크게 줄였기 때문이다. 텐서 초기화 후 반복적인 추론 연산을 수행했을 때도 컴포저블 CXL은 기존 시스템 대비 전체 처리량을 약 3.32배 향상시켰다. 구체적으로 텐서 초기화와 추론 단계에서 각각 2.71배와 3.51배의 속도 개선을 보였다(그림 35d). 이 성능 개선은 전자상거래나 스트리밍 서비스 등 상용 플랫폼의 개인화된 콘텐츠 제공 속도를 높여 사용자 경험을 크게 향상시킬 수 있다.

MPI 기반 과학 응용 사례: CXL을 활용한 메모리 공유 성능 평가. MPI 기반의 과학 컴퓨팅 프로그램은 노드 간 빈번한 데이터 통신과 동기화 과정에서 발생하는 지연으로 인해 기존 네트워크 기반 아키텍처 [387–389]에서 성능 확장성에 한계가 있다. 본 연구는 주로 AI 워크로드에 중점을 두고 있으나, 컴포저블 CXL 인프라를 통해 직접 메모리 공유 방식을 활용할 때의 이점을 확인하기 위해 대표적인 MPI 과학 응용 프로그램을 평가하였다. 예를 들어, 입자-셀(PIC, Particle-In-Cell [390–392]) 플라즈마 시뮬레이션과 전산유체역학(CFD, Computational Fluid Dynamics [393–395]) 시뮬레이션은 노드 간 데이터 교환과 상태 동기화가 빈번하여 분산 AI 환경의 통신 패턴과 유사하다.



(a) 초반 (t_0). **(b)** 중반 (t_1). **(c)** 후반 (t_2). **(d)** 실행 속도 개선.

|그림 37| MPI 활용 사례: CFD 시뮬레이션.

본 연구에서는 위에서 예제로 제시된 두 가지 대표적인 MPI 시나리오를 평가하였다. 첫 번째로 그림 36와 같이 PIC 프레임워크인 WarpX [390]를 이용한 플라즈마 시뮬레이션을 수행했다. 수억 개의 입자가 여러 계산 노드에 분산되어 상호작용하는 시나리오에서, 기존의 인피니밴드 기반 RDMA 시스템은 소프트웨어 개입 및 장치 간 데이터 이동으로 인해 높은 오버헤드를 초래하였다. 이에 반해, 본 연구에서 사용한 컴포저블 CXL 기반 인프라는 호스트 CPU가 입자 데이터를 동적으로 구성 가능한 CXL.cache 공유 메모리에 직접 저장하여, 다른 노드들이 소프트웨어 프로토콜을 거치지 않고 즉시 데이터에 접근할 수 있었다. 이를 통해 그림 36d에서 분석된 것과 같이 기존 RDMA 시스템 대비 계산과 통신에서 각각 1.62배, 6.46배의 성능 향상을 달성했다.

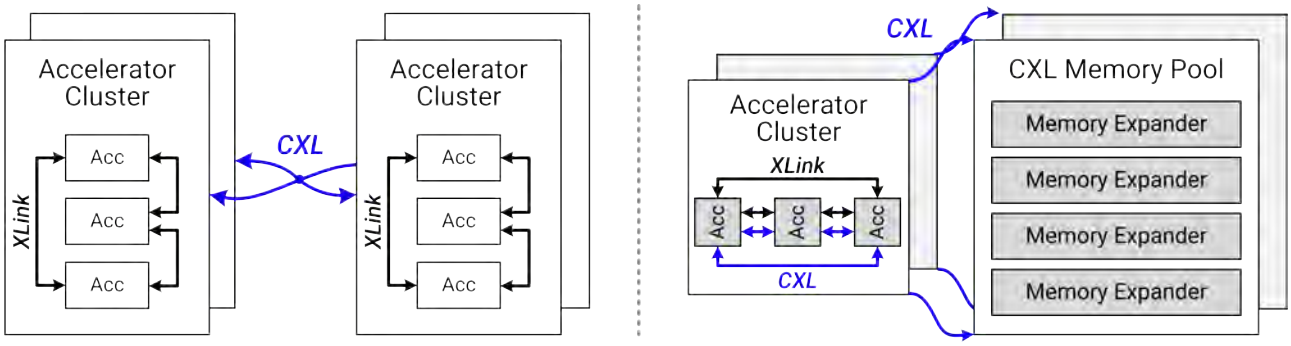
두 번째 사례로 그림 37에 표현된 CFD 시뮬레이션을 수행하였다. 이 시나리오에서는 도메인 간 유체 상태를 빈번히 동기화할 때 기존 RDMA 기반 네트워크에서 상당한 지연이 발생하였다. 그러나 컴포저블 CXL 인프라는 유체 시뮬레이션 상태 데이터를 공유 메모리에 직접 저장하고 CPU가 이를 바로 접근할 수 있도록 함으로써 네트워크 기반 동기화 작업을 명시적으로 없앴다. 이는 데이터 일관성과 캐시 일관성이 CXL.cache에 의해 자동 관리되었기 때문이다. 그 결과, 그림 37d에서 볼 수 있듯, 기존 시스템 대비 계산 시간은 약 1.06배, 통신 시간은 약 3.57배 단축되었다.

비록 오픈소스 기반의 하드웨어에 CXL 컨트롤러와 스위치를 통합한 프로토타입 시스템으로 평가한 결과이지만, 이러한 MPI 기반 과학 응용 사례 실험을 통해서 우리는 CXL 기반의 인프라가 HPC 환경에서 명시적인 네트워크 작업을 제거하여 지연과 소프트웨어 오버헤드를 크게 줄일 수 있음을 확인하였다. 또한 CXL 기반 인프라가 MPI 기반 과학 응용, 병렬 작업 처리 시 자원 분리와 메모리 공유를 통해 데이터 관리를 단순화하고, 분산 환경에서의 확장성과 운영 효율성을 향상할 수 있음을 확인할 수 있었다. 이러한 개선 효과는 기후 모델링, 천체 물리학, 핵융합 연구와 같이 다른 대규모 과학 응용 분야 등에서도 매우 유용할 것으로 기대된다.

6. CXL을 넘어: 하이브리드 링크 아키텍처를 활용한 AI 자원 연결 최적화

CXL은 메모리 확장과 일관된 데이터 공유라는 중요한 문제를 해결하지만, 일부 가속기 중심의 워크로드에서는 가속기 간의 데이터 교환 성능을 더욱 높이기 위해 다른 인터커넥트 기술과 결합하여 사용하는 것이 데이터센터 전체 효율성을 극대화하는 데 도움이 될 수 있다. 대표적으로 가속기 간 통신에 최적화된 인터커넥트 기술로는 UALink와 NVIDIA의 NVLink가 있으며, 본 보고서에서는 이 둘을 통칭하여 "엑스링크(XLink)"라고 부르도록 하겠다.

XLink 기술은 가속기 간의 데이터 전송 속도를 최적화하는 P2P 방식의 직접적인 연결을 제공하여 밀집된 가속기 클러스터 내에서 데이터 전송 성능을 크게 향상시킨다. 다만 XLink는 CXL과 달리 프로토콜 수준의 캐시 일관성이나 메모리 풀링과 같은 기능을 지원하지 않는다. 대신 XLink는 단일 홉의 인터커넥트 구조를 사용하여 낮은 지연과 높은



(a) 가속기 중심 클러스터. **(b)** 계층적 메모리 아키텍처.

|그림 38| 하이브리드 인터커넥트 구조(CXL-over-XLink)의 개괄.

효율성을 갖춘 가속기 간 데이터 전송에 특화되어 있다. UALink와 NVLink는 단일-홉 토폴로지 정책 등 가속기 연결에 대한 유사한 설계 철학을 갖고 있지만, 세부적인 구현 방식에는 차이가 있다. UALink는 대용량 데이터 전송에는 물리 계층으로서 이더넷 기반 기술을 사용하는 반면, NVLink는 텐서 전송 및 GPU 간 그래디언트 동기화와 같은 중소 규모 데이터 교환에 특화된 NVIDIA의 독자적인 전기적 신호 전송 기술을 활용한다.

그림 38a와 그림 38b에서 볼 수 있듯이, 본 보고서에서는 CXL과 XLink의 상호보완적인 장점을 결합하여 전체 시스템 성능을 최적화할 수 있는 "하이브리드 인터커넥트 구조(CXL-over-XLink)"를 제시한다. 또한, CXL-over-XLink를 구현하기 위한 두 가지 설계 방법을 제시한다. 첫 번째는 "가속기 중심 클러스터" 방식으로, 가속기 간의 빠른 데이터 교환을 극대화하는 구조이다. 두 번째는 "계층적 메모리 아키텍처" 방식으로, 대규모 데이터를 효율적으로 처리하기 위해 분리된 메모리 풀을 적극적으로 활용하는 설계이다. 두 설계 구조 모두 CXL과 XLink의 장점들을 결합, LLM과 같은 대규모 워크로드를 실행하는 데 가속기 간 호환성을 최대화하면서 연산과 메모리 요구량 모두를 만족시킬 수 있도록 설계되었다.

구체적으로, XLink는 단일 홉의 클로스 토폴로지를 사용하여 가속기 간의 데이터 교환을 최적화하며, 텐서 데이터 전송이나 그래디언트 동기화와 같은 지연에 민감한 작업에서 뛰어난 성능을 제공한다. 그러나 XLink의 단일-홉 클로스 토폴로지는 가속기의 수가 늘어날 경우 연결성 확장에 한계가 있다. 반면, CXL은 다단계 스위치 캐스케이딩을 지원하여 보다 높은 확장성을 제공하며, 데이터센터 전반에서 다양한 토폴로지 구성을 가능하게 한다. 또한 CXL은 프로토콜 수준에서 캐시 일관성 메모리 풀링을 제공하기 때문에 KV 캐싱 및 RAG와 같이 메모리 집약적이고 동적인 워크로드에 효과적이다. 더 나아가, CXL은 노드 간 데이터 공유를 효율적으로 지원하여 불필요한 데이터 이동을 최소화하고 메모리 활용도를 크게 향상시킨다.

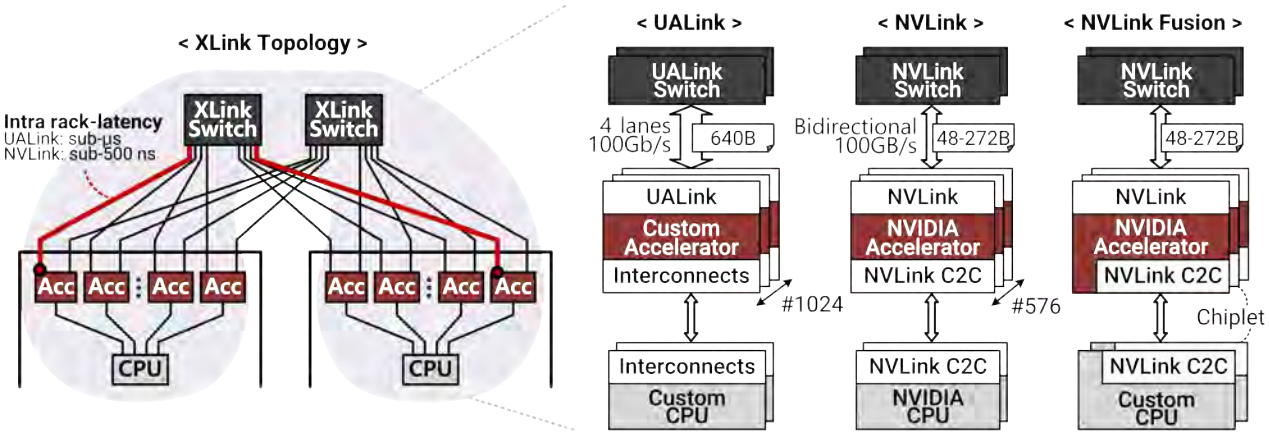
또한 컴포저블 자원 분리 개념을 활용하면, 계산 자원과 메모리를 물리적으로 분리하여 독립적인 확장과 유연한 관리가 가능하다. XLink로 연결된 컴퓨트 노드와 CXL 기반의 메모리 풀을 함께 운용하면, 연산 집약적인 학습 워크로드와 지연이 중요한 추론 워크로드 사이를 신속하게 전환하는 운영상의 민첩성을 확보할 수 있다. 또한 메모리 자원을 성능과 용량 요구에 따라 계층적으로 배치하여 자원 활용도를 최적화할 수 있다.

본 섹션에서는 다양한 인터커넥트 기술의 이해를 돕기 위해 XLink 기술을 간단히 먼저 설명하고, UALink와 NVLink 각각의 주요 특성과 최적화 방안을 중점적으로 논의한다. 이어서 CXL과 XLink를 결합한 하이브리드 아키텍처 전략에 대해 자세히 소개하며, 두 기술의 상호보완적인 강점이 현대 AI 데이터센터의 다양한 워크로드 요구를 어떻게 효과적으로 충족시킬 수 있는지 구체적으로 제시한다.

6.1. 가속기 중심 인터커넥트 개요: UALink와 NVLink

울트라 엑셀러레이터 링크 (UALink). UALink는 데이터센터 내 가속기 간의 통신 속도를 높이기 위해 설계된 인터커넥트 기술이다 [70, 396, 397]. CXL이 메모리의 독립적인 확장, 일관된 메모리 관리, 통합된 메모리 풀 구성과 같은 메모리 중심 기능에 초점을 맞추고 있는 반면, UALink는 가속기 사이에서 높은 대역폭으로 직접 데이터를 주고받는 데 주력한다. 따라서 가속기 간에 데이터 교환량이 많고 빠른 동기화가 필요한 워크로드에서 특히 효과적이다. 일반적인 4레인 구성 기준으로 UALink는 포트당 최대 100GB/s의 높은 대역폭을 제공할 수 있다.

2025년 초에 도입된 UALink 1.0 [69]은 GPU 중심의 NVLink와 비슷한 특징을 갖고 있지만, 특정 GPU 제조사에



|그림 39| 가속기 중심 인터커넥트.

의존하지 않고 다양한 가속기를 지원하는 개방형 표준(Open Standard)으로 설계되었다. UALink는 단일-홉 클로스 스위치 토폴로지를 사용하여 가속기 간에 직접 연결된 저지연 통신 경로를 구축하며, 최대 1,024개의 가속기를 하나의 클러스터로 연결할 수 있다. 그림 39에 나타난 것처럼, 이 단순한 구조를 통해 랙 내부의 통신 지연을 1마이크로초(μs [70]) 이하로 매우 낮게 유지할 수 있다. 이러한 단순화된 토폴로지와 높은 확장성 덕분에, UALink는 밀접하게 연결된 가속기 클러스터에서 필요한 빈번한 데이터 교환과 빠른 동기화 요구를 효과적으로 만족할 수 있다.

UALink는 특히 큰 용량의 데이터 전송을 빠르게 처리하기 위해 640바이트의 데이터 플릿(Flit) 단위를 사용한다. 또한 기본적인 명령어 기반의 메모리 접근 메커니즘도 제공하지만, 이는 주로 관리 및 제어를 위한 보조적인 기능으로 활용되며, UALink의 주된 목적은 고속의 대규모 데이터 전송에 있다. 또한 UALink는 캐시 일관성이나 메모리 풀링 같은 일관성 유지 기능을 기본적으로 제공하지 않으며, 명시적으로 비일관성(Non-Coherent) 프로토콜로 설계되어 있다. 이는 빈번한 데이터 공유와 일관성 메모리 관리, 그리고 자원의 분리를 주력으로 하는 CXL과 근본적으로 다른 특징이다.

UALink는 이더넷 기반의 인터커넥트 토폴로지를 채택하여 All-Gather 작업과 같은 분산 시스템에서 흔히 사용되는 집합적 통신 패턴에 특히 유리하다. 이러한 통신 방식은 과거부터 분산 시스템에서 널리 사용되었으며, 최근의 다중 GPU 및 가속기 환경에서도 중요성이 높아지고 있다 [8, 40, 386, 398, 399]. 빠르면서도 정확한 데이터 전달을 위해, UALink는 가속기 간의 빠른 동기화에 맞춘 특수한 프레임 구조와 프로토콜 오버헤드를 최소화하는 하드웨어 기반의 동기화 메커니즘을 적용하고 있다 [69, 397].

엔브이링크(NVLink)와 엔브이링크 퓨전(NVLink Fusion). NVLink는 GPU 간 데이터 교환을 최적화하기 위해 NVIDIA가 개발한 인터커넥트 기술로, 높은 대역폭과 낮은 지연 성능을 제공한다. NVLink는 UALink보다 먼저 개발되었으며, 기존 PCIe 기반 인터페이스 대비 GPU 간 데이터 전송 성능을 크게 향상시켰다. 2014년 처음 공개된 이후 여러 세대를 거쳐 진화했으며, 가장 최신 버전인 NVLink 5.0은 2024년에 발표되었다. NVLink는 딥러닝 학습과 HPC 워크로드에서 본인들의 GPU 장치가 탁월한 성능을 발휘하도록 최적화되어 있다.

구체적으로, NVLink 5.0은 링크당 50GB/s의 단방향 대역폭을 제공하며, 양방향 기준 최대 100GB/s의 총 대역폭을 지원한다 [73]. NVLink는 엔비디아의 자체 크로스바(Crossbar) 스위치 기술인 NVSwitch를 통해 구축되며, 소규모 GPU 클러스터(NVLink72 [244, 262, 263] 등)에서부터 보다 큰 규모의 GPU 환경(NVLink576 등)에

이르기까지 다양한 규모의 구성을 지원한다. 다만, NVLink는 주로 단일 GPU 노드나 클러스터 수준에서의 최적화를 목표로 하고 있어 랙 단위 이상의 대규모 환경 확장에는 제한적이다. 또한 UALink와 마찬가지로 단일-홉 클로스 토폴로지를 사용하여, All-Reduce나 All-Gather와 같은 집합적 연산에서 발생하는 통신 지연을 최소화한다. 최신 버전인 NVLink 5.0의 지연 시간은 500나노초(ns) 이하로 매우 짧다 [400].

UALink와의 중요한 차이점 중 하나는, NVLink가 48바이트에서 272바이트의 비교적 작은 크기의 플릿을 사용한다는 점이다[7] [401]. 이는 중소 규모의 텐서나 그래디언트 데이터 전송에 최적화되어 있다. NVLink는 노드 간 메모리 영역을 부분적으로 통합하여 소프트웨어 프로그래밍의 복잡성을 낮추고 오버헤드를 줄이지만, 완전한 캐시 일관성을 기본적으로 제공하지는 않는다. 또한 NVLink는 기존까지 엔비디아의 제품군 내에서만 제한적으로 사용 가능하여, 다른 제조사의 장비와의 통합은 어렵다는 단점이 있었다.

이러한 한계를 개선하기 위해 최근 엔비디아는 NVLink 퓨전(NVLink Fusion [74, 75])을 도입하였다. NVLink 퓨전은 CPU, NPU [402–404], 그리고 AI 전용 프로세서 [405–414] 등 다양한 외부 프로세서와 GPU 간의 연결성을 크게 향상시켜, 상호 운용성을 높였다. NVLink 퓨전은 크게 두 가지로 구성되어 있다. 첫째는 캐시 일관성을 지원하는 칩 간(C2C) 단거리 인터페이스로서 외부 프로세서와 GPU 간의 직접적인 데이터 공유를 가능하게 하는 코히어런트 IP(Coherent IP)이다. 둘째는 칩렛(Chiplet) 기반의 구현 방식을 통해 CPU와 GPU 간의 통신 성능을 추가적으로 최적화하는 것이다.

NVLink 퓨전은 기존 NVLink의 강점인 높은 대역폭과 저지연 성능을 유지하면서도, CPU와 GPU 간의 공유 메모리 접근을 더욱 유연하고 효율적으로 만들어준다. 그러나 NVLink 퓨전 역시 시스템 내에 최소 하나 이상의 엔비디아 컴포넌트가 포함되어야 한다는 정책적 제약을 가지고 있다 [74, 75]. 따라서 현재로서는 다양한 제조사의 제품을 조합하여 사용하는 이기종 환경(Heterogeneous Environment)에서의 자원 분리나 제조사 중립적 컴포저빌리티에 사용되기가 어렵다.

CXL과 XLink 기술의 주요 특성 비교. 표 3에서는 본 보고서에서 다룬 세 가지 주요 인터커넥트 기술인 CXL, UALink, NVLink 간의 특징을 정리하여 비교하였다.

앞서 설명한 바와 같이, CXL은 주로 컴퓨팅과 메모리 자원을 물리적으로 분리하고, 일관된 메모리 풀 및 캐시 일관성을 지원하는 데 중점을 둔다. 표에서 볼 수 있듯이 CXL은 메모리 자체를 다루기 때문에, 지연 시간이 다른 어떤 인터커넥트 기술보다 짧다. 더욱이, 캐시 일관성을 지원하기 때문에 메모리 접근 시 가속기 내 캐시에서 바로 데이터가 서비스되어 지역성을 가지는 데이터에 대해서 외부 전송이나 접근 자체를 원천적으로 막아, 성능을 극대화할 수 있다는 장점이 있다. 가속기(Type 1 또는 Type 2 장치)에 대해서 최대 256개까지만 연결할 수 있지만, 메모리 장치처럼 연결하면 최대 4,096개의 엔드포인트를 하나의 인터커넥트 네트워크 안에 수용할 수 있으며, 이러한 장치 연결을 위한 실질적인 PBR 라우팅, 그리고 스위치 케스케이딩을 지원한다. 이와 같은 특장점에 의해, 자원의 확장성과 캐시 일관성 유지, 메모리 자원의 유연한 할당과 같은 부분에서 강점을 가진다. 따라서 CXL은 빈번한 동기화 통신 및 메모리 사용량이 많은 워크로드와 자원을 유연하게 구성해야 하는 컴포저블 환경에 적합하다.

이에 비해, XLink 기술들은 가속기 간의 직접적이고 빠른 데이터 전송을 최우선으로 하며, 높은 대역폭의 연결성을 제공하는 데 집중한다. 하나의 데이터 전송에 대한 지연 시간은 CXL보다 느린 편이지만 플릿의 크기가 큰 편이거나

[7] NVLink에서 실제 플릿의 크기는 16바이트(128비트)이지만, 하나의 패킷은 헤더 플릿 1개와 최대 16개의 데이터 플릿으로 구성된다. 최소 전송 단위는 2개의 데이터 플릿(32바이트)이며, 최대는 16개의 데이터 플릿(256바이트)로 구성되어 전체 패킷 크기는 48바이트에서 272바이트 사이 범위를 가진다. 본 절에서는 이러한 구성 형식을 따르는 전체 패킷을 NVLink 플릿이라고 정의한다.

|표 3| CXL, UALink, NVLink의 기술 사양 비교.

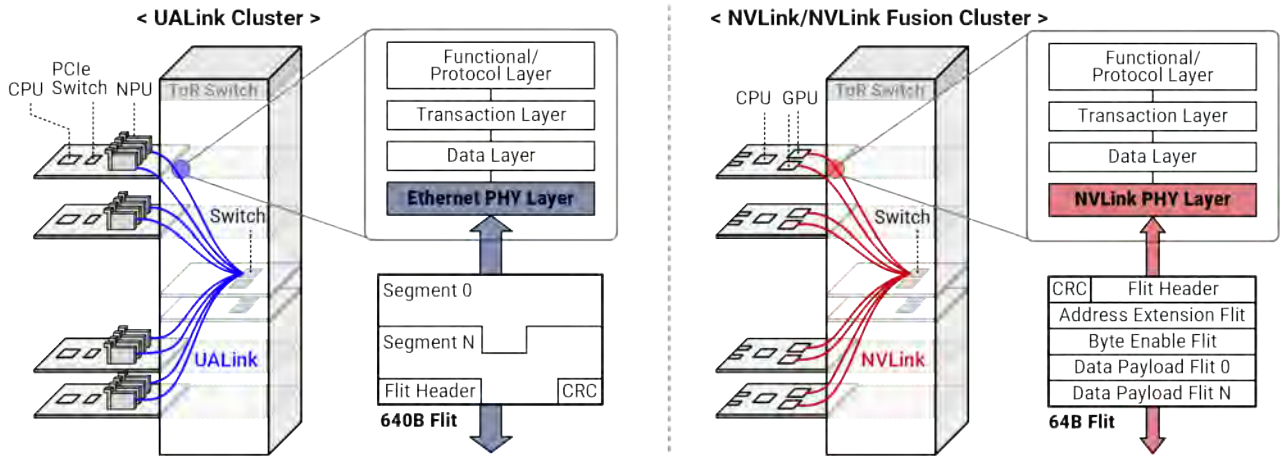
항목	CXL 3.0	UALink 1.0	NVLink 5.0
단방향 대역폭 (GB/s)	128 (링크당 16레인, PCIe 6.0)	100 (링크당 4레인)	50 (링크당 2레인)
지연 시간	수백 나노초 (일반적으로 100–250 ns)	1 마이크로초 이하 (랙 내 기준)	500 나노초 이하 (랙 내 기준)
플릿 크기	256B (PBR), 68B (HBR)	640B	48B∼272B
캐시 일관성	지원 (하드웨어 수준)	미지원	미지원 (NVLink C2C만 지원)
메모리 풀링	지원	미지원 (UALink로 연결된 가속기 내에서만 지원)	미지원 (NVLink로 연결된 가속기 내에서만 지원)
토폴로지	P2P, 스위치 기반 패브릭 (다양한 토폴로지)	P2P, 스위치 기반 패브릭 (단일-홉 클로스만 지원함)	P2P, 스위치 기반 패브릭 (단일-홉 클로스만 지원할 수 있음)
확장성	최대 4096개 장치	최대 1024개 가속기	최대 576개 GPU
일반적인 배치 규모	랙 또는 다중 랙 규모	랙 내 클러스터 규모	GPU 노드 또는 GPU 클러스터 규모
용도 / 주요 워크로드	메모리 분리, 캐시 일관성 메모리 풀링	가속기 간 집합적 통신	GPU 텐서 교환, 그래디언트 동기화
컨소시엄	CXL 컨소시엄	UALink 컨소시엄	NVIDIA
상호운용성	공개 산업 표준	이더넷-기반 개방성	독점 (NVLink 퓨전을 통한 부분적 개방)
최초 출시 (연도)	CXL 1.0 (2019)	UALink 0.49 (2024)	NVLink 1.0 (2016)
현재 버전 (연도)	CXL 3.0 (2022), CXL 3.2 (2024)	UALink 1.0 (2025)	NVLink 5.0 (2024), Fusion (2025)

특정 GPU나 가속기 장치에 최적화되어 높은 대역폭을 제공할 수 있다. 특히 UALink는 이더넷 기반의 네트워크를 사용하여 대규모 데이터를 효율적으로 전송하며, 여러 가속기 간 빈번한 데이터 교환 및 집합적 통신 패턴에 최적화되어 있다. 반면 NVLink는 GPU 중심의 워크로드에서 중소 규모의 텐서 데이터나 그래디언트를 빠르게 교환하는 데 특화되어 플릿의 사이즈가 작고 대역폭과 지연시간의 최적화가 이루어져 있는 것으로 알려져 있다. 다만 이러한 데이터 이동은 기본적으로 데이터 공유를 통한 병렬 처리보다는 복사를 통한 집합 통신과 분산 처리를 가정하기 때문에 두 프로토콜 모두 하드웨어 수준의 캐시 일관성은 지원하지 않는다.

따라서 본 보고서에서 데이터센터 아키텍처로써 제안하는, XLink 기술의 효율적인 가속기 통신과 CXL의 유연한 메모리 풀링 및 캐시 일관성을 결합한 하이브리드 아키텍처는 상호 보완적이며 강력한 솔루션을 제공할 수 있다. 이를 통해 UALink 또는 NVLink로 상호 연결된 가속기들이 CXL로 구성된 메모리 자원 및 일관성 프로토콜을 함께 활용할 수 있도록 만들 수 있으며 기존 데이터 병렬 접근 방식보다 훨씬 넓은 범위에서 스케일업 도메인을 구축하고 더욱 높은 자원 활용도와 효율성을 달성할 수 있다.

6.2. 통합 가속기 중심의 CXL-over-XLink 슈퍼클러스터 아키텍처

대규모 AI 워크로드의 다양한 요구사항을 충족하려면 개별 클러스터를 넘어 클러스터 간의 효율적인 데이터 통신이 필요하다. 참고로 여기서 클러스터란, 앞서 소개된 데이터센터 구조에서 랙 규모의 다중 가속기 시스템을 의미한다. 이러한 구조에서 단일 클러스터 내부에서는 P2P 연결을 위해 간단한 인터커넥트 구조만으로 충분하지만, 여러 클러스터를 연결할 때는 보다 유연하고 다양한 형태로 확장 가능한 토폴로지가 필요하다. 또한, 데이터센터 전체의 이종 간 자원을 효율적으로 공유하고 컴포저블하게 구성할 수 있어야 한다. CXL은 다단계 스위치 구조를 활용하여



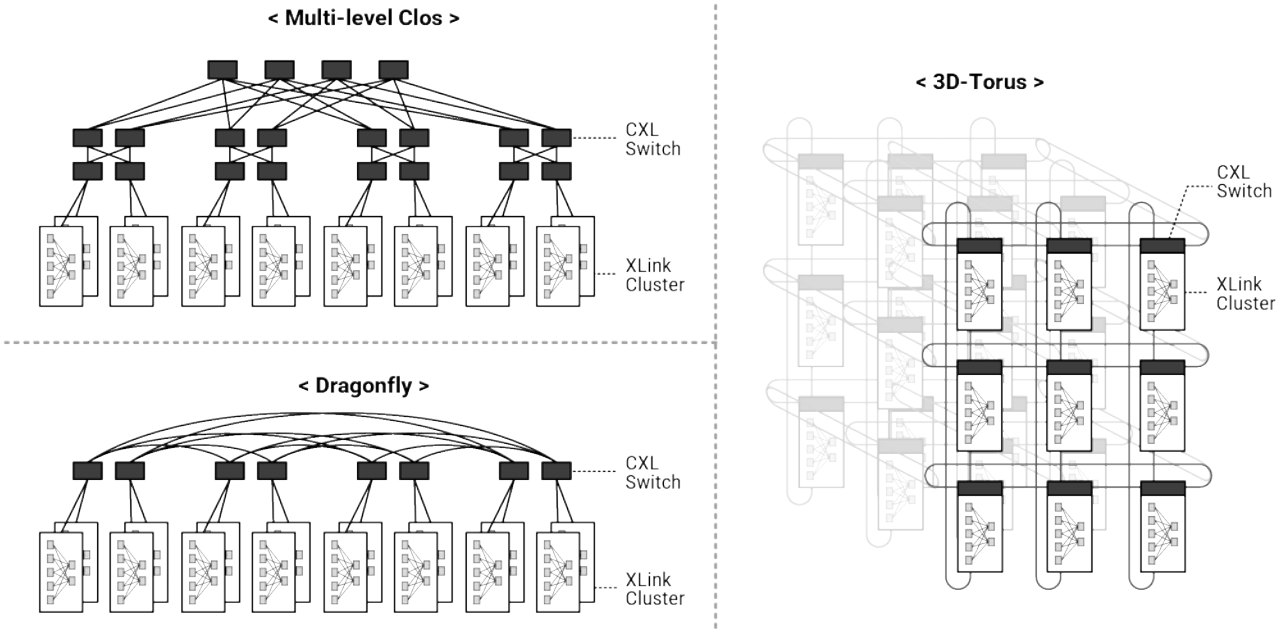
|그림 40| 가속기 중심 클러스터 내부 설계.

이러한 대규모 클러스터 간 자원 공유를 효과적으로 지원할 수 있다.

본 절에서는 가속기 중심의 작업에 최적화된 CXL-over-XLink로 구성된 "슈퍼클러스터" 아키텍처를 제안하고 설명한다. 이 슈퍼클러스터는 여러 가속기 클러스터가 CXL 기반의 계층적 네트워크로 연결된 구조이다. 개별 클러스터 내부에서는 NVLink 또는 UALink를 사용하여 가속기 간의 직접적이고 빠른 데이터 통신을 수행하며, 각 클러스터의 특징과 세부 구성에 대해 아래에서 자세히 다룬다.

UALink 및 NVLink 기반의 가속기 클러스터 내부 설계. CXL-over-XLink 기반의 슈퍼클러스터 아키텍처에서 NVLink와 UALink는 개별 클러스터 내부의 인터커넥트 기술로 활용된다. 앞서 설명한 바와 같이 이 인터커넥트 기술들은 유사한 기본 설계 원칙을 공유하며, 그림 40에 보여진 것처럼 주로 소규모 가속기 클러스터에서 가속기 간 통신을 최적화한 단일-홉 클로스 스위치 토폴로지를 사용하도록 구성할 수 있다. NVLink는 복수의 NVSwitch를 통해 최대 72개의 GPU를 상호 연결할 수 있으며, 각 노드 내의 CPU는 NVLink C2C 인터페이스를 통해 GPU와 연결된다. 이와 유사하게, UALink 또한 클러스터 및 랙 내의 가속기 연결을 명확히 목표로 하며, 애플리케이션별 요구사항에 따라 CPU를 가속기에 직접 연결할 수 있다. UALink 기반 클러스터의 가속기들은 오직 UALink 스위치를 통해서만 통신하며, 이론적으로 최대 1,024개 가속기까지 지원 가능한 단일-홉 클로스 토폴로지를 형성할 수 있다. 이러한 높은 확장성은 논리 규모(Logic Size)가 작은 NPU와 같은 AI 전용 가속기에 유리하다. 그러나 GPU와 같이 논리 규모가 큰 가속기의 경우, 노드당 장착 가능한 가속기 수가 제한되기 때문에(예를 들어 GB200/300은 노드당 GPU가 두 개로 제한됨), 실제 랙 단위의 배치 규모는 NVLink의 구성과 유사한 수준(약 72개 가속기)이다. 한편 UALink에서는 CPU와 가속기 간 연결이 일반적으로 PCIe 스위치를 통해 이루어지지만, NVLink C2C와 유사한 방식으로 UCIe [415]와 같은 단거리 C2C 기술을 활용할 수도 있다.

이와 같이 구성된 가속기 클러스터는 일반적으로 단일 클러스터 내에서 NVLink와 UALink를 혼합 사용하기보다는, 하나의 XLink 기술만을 채택할 가능성이 높다. 이는 NVLink와 UALink 간의 근본적인 기술적 차이와 상호운용성 제약 때문이며, 구체적으로 두 인터커넥트 기술이 사용하는 물리 계층(PHY) 및 링크 데이터 포맷의 차이에서 비롯된다. 앞서 언급된 것처럼, NVLink는 NVIDIA 고유의 고속 PHY 인터페이스를 사용하며 비교적 작은 48바이트에서 272바이트 크기의 플릿을 사용하는 반면, UALink는 이더넷 기반 PHY 인터페이스에 크기가 큰 640바이트 플릿을



|그림 41| CXL-over-XLink 기반 슈퍼클러스터 구성 예시.

채택한다. 따라서 플릿 포맷, 프로토콜 운영 방식, 물리 계층의 차이 등으로 인해 NVLink와 UALink 하드웨어를 동일 클러스터 내에서 함께 사용하는 것은 실질적으로 어렵다. 무엇보다도, NVLink는 최소한 하나의 NVIDIA 컴포넌트(예: NVIDIA GPU)를 요구하기 때문에 완전히 서드파티(Third-Party) 가속기만으로 구성하는 데도 심각한 제한이 있다.

이에 따라 CXL-over-XLink 슈퍼클러스터 내에서 NVLink와 NVSwitch 기반으로 구성된 가속기 클러스터는 주로 NVIDIA GPU로 이루어지며, GPU가 효과적으로 처리하기 어려운 특수 연산을 최적화한 전용 가속기들을 함께 사용할 수 있다. 예를 들어 NVIDIA GPU가 배치된 데이터센터는 분기(Branch)가 많은 연산(트리 기반 모델, 조건 로직 등), 불규칙한 제어 흐름을 가지는 작업, 희소하거나 불규칙한 메모리 접근 패턴을 요구하는 워크로드(그래프 처리, 희소 행렬 연산 등), 혹은 실시간 처리와 같이 지연에 민감한 연산에 특화된 가속기를 추가할 수 있다. 이러한 연산들은 고도의 병렬 처리에 최적화된 GPU 아키텍처와는 효율적으로 맞지 않기 때문이다. 이처럼 다양한 특성의 가속기들을 NVLink 기반 클러스터에 통합함으로써 데이터센터는 애플리케이션의 다양한 요구를 수용하면서도 클러스터 내부 통신 성능을 최적화할 수 있다.

반면, UALink 기반 클러스터는 주로 NVIDIA가 아닌 AMD GPU나 메타의 MTIA [405, 406], 아마존의 트레이니엄(Trainium [407, 408]) 및 인퍼렌시아(Inferentia [409, 410]), 마이크로소프트의 마이아(Maia [411, 412]), 인텔의 가우디(Gaudi [413, 414]) 등과 같은 AI 전용 가속기들로 구성된다. UALink의 개방적이고 벤더 중립적인 아키텍처는 다양한 가속기 조합을 가능하게 하며, 독점 인터페이스에 대한 의존성 없이 고성능의 클러스터 내부 통신을 지원한다. 각 인터커넥트의 상호운용성 특성과 아키텍처적 강점에 따라 전략적으로 배치 방식을 결정하면, 이기종 가속기 환경에서 클러스터 내 성능, 계산 처리량 및 자원 효율성을 최적화할 수 있다.

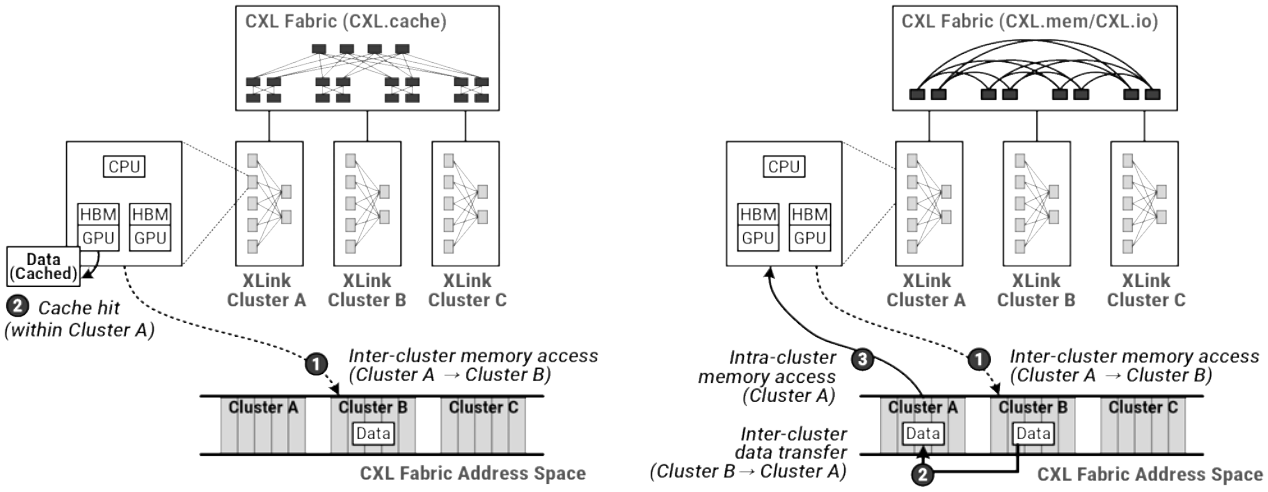
CXL을 활용한 클러스터 간 확장 가능한 통신. CXL-over-XLink 기반의 슈퍼클러스터는 다수의 XLink 기반 가속기 클러스터를 확장 가능한 CXL 네트워크로 통합하여 하나의 대규모 다중 가속기 시스템으로 구성된다. 본 기술 보고서

에서 제시하는 이 하이브리드 인터커넥트 패브릭 방식은 가속기 간 낮은 지연과 효율적인 데이터 전송을 유지하면서 현대 AI 데이터센터의 다양한 워크로드 요구를 효과적으로 수용할 수 있는 환경을 제시한다. 구체적으로, XLink가 가속기 클러스터 내부의 빠른 데이터 교환을 지원한다면, CXL은 클러스터 간에 유연하고 확장 가능한 스위치 기반의 네트워크를 형성하여 메모리를 일관되게 공유할 수 있게 해준다. 특히, XLink가 제한적인 단일-홉 클로스 구조를 사용하는 반면, CXL은 계층적이고 확장 가능한 구조를 통해 여러 클러스터가 통합된 메모리 풀을 구성할 수 있도록 지원한다. 이를 통해 외부 메모리나 SSD와 같은 저장 장치의 사용을 최소화하여 성능 이득을 최대화하고 가속기 간 한정된 자원의 유연성과 활용도를 높일 수 있다. 또한, CXL은 UALink와 NVLink 간의 호환성 문제를 해결하는 중재자 역할을 하며, 다양한 이기종 가속기 클러스터가 하나의 대규모 아키텍처 내에서 원활히 상호작용 하도록 도울 수 있다.

그림 41은 여러 개의 UALink 및 NVLink 기반 클러스터가 계층적 CXL 스위치를 통해 하나의 슈퍼클러스터를 구성할 때 이 스위치들을 통해 만들어질 수 있는 패브릭 아키텍처를 보여준다. CXL은 PBR 라우팅과 스위치 캐스케이딩을 통해서 멀티-레벨 클로스, 3D-토러스, 드래곤플라이 토폴로지와 같은 다양한 연결 방식으로 여러 데이터센터 요구사항을 만족시킬 수 있도록 구성 가능하다. 또한, CXL은 대규모 AI 데이터센터 운영 중에도 장치나 클러스터를 핫플러그 방식으로 자유롭게 추가하거나 제거할 수 있는 유연성을 제공한다. 따라서 워크로드의 특성에 따라 네트워크를 유연하게 조정하고, 다양한 유형의 가속기나 메모리 장치 그리고 연산 장치들을 스케일업 환경 내에서 하나의 슈퍼클러스터로 통합할 수 있다.

이 CXL-over-XLink 구조의 또 다른 주요 장점은 클러스터 간 프로토콜 수준의 캐시 일관성이다. CXL이 제공하는 캐시와 관련된 하위 프로토콜(CXL.cache) 메커니즘을 통해 클러스터 내부의 가속기들은 자신 이외 다른 가속기의 메모리와 외부 클러스터에 존재하는 원격 메모리 자원을 소프트웨어 없이 직접 명령어 수준에서 일관되게 접근할 수 있다. 이러한 방식은 가속기가 외부 메모리 없이 다수 가속기의 로컬 메모리 등을 하나로 모으고 모두가 통일된 메모리 주소 공간을 볼 수 있게 해 주며, 데이터는 각 가속기 온칩 캐시에서 바로 끌어올 수 있도록 하여 기존의 노드 간 메모리 접근 방식에서 흔히 발생하는 지연과 데이터 이동 오버헤드는 물론, 지역성을 가진 데이터나 공유 데이터는 가속기 본인의 캐시에서 바로 서비스할 수 있도록 하여 성능을 최대화할 수 있다. 또한, CXL은 전용 메모리 접근에 관련된 하위 프로토콜(CXL.mem)과 고속의 데이터 트랜잭션에 관련된 하위 프로토콜(CXL.io) 인터페이스를 모두 제공하여, 작은 크기의 명령어 수준 데이터 전송과 큰 데이터 블록의 전송 모두에 대응할 수 있고, 장치 간 직접 연결로 CPU의 개입이 전혀 없이도 데이터를 관리할 수 있도록 프로토콜을 수정할 수도 있다. 결과적으로, 그림 42에서처럼, 하이브리드 인터커넥트 구조를 사용하여 가속기 간 데이터 이동을 최소화할 수 있으며, 분산된 자원을 통한 연산 가속화를 지원하여 메모리 집약적인 워크로드에서도 높은 성능을 유지할 수 있도록 도와줄 수 있다.

마지막으로, CXL-over-XLink는 슈퍼클러스터 내에서 기존 집합 연산 등에 의한 데이터 이동 방식의 효율성을 한층 더 향상시킬 수 있다. 특히, 기존 시스템에서 데이터 복사나 명시적 동기화 작업으로 인해 많은 오버헤드를 유발하는 브로드캐스트(Broadcast), 스캐터/개더(Scatter/Gather), All-Reduce와 같은 집합적 연산의 처리 방식을 근본적으로 개선할 수 있다. 이는 기존의 명시적 동기화와 중복 데이터 복사로 인해 발생하던 오버헤드를 원천적으로 없앨 수 있기 때문이다. 분산된 가속기들이 CXL을 통해 캐시 일관성을 보장하는 메모리에 접근을 수행하면, 데이터 이동이 하드웨어 수준에서 자동으로 관리되어 분산된 자원을 통합된 메모리 풀로 사용할 수 있다. 이러한 접근법은 성능을 높일 뿐 아니라 AI 모델 개발과 관리를 단순화하는 장점도 제공한다. 예를 들어, 개발자는 가속기 커널(Kernel)을 프로그래밍할 때 복잡한 동기화 및 데이터 이동 코드를 명시적으로 관리할 필요 없이, 오직 연산 작업 자체에만



(a) 프로토콜 수준의 캐시 일관성. **(b)** 명령어 수준의 데이터 전송.

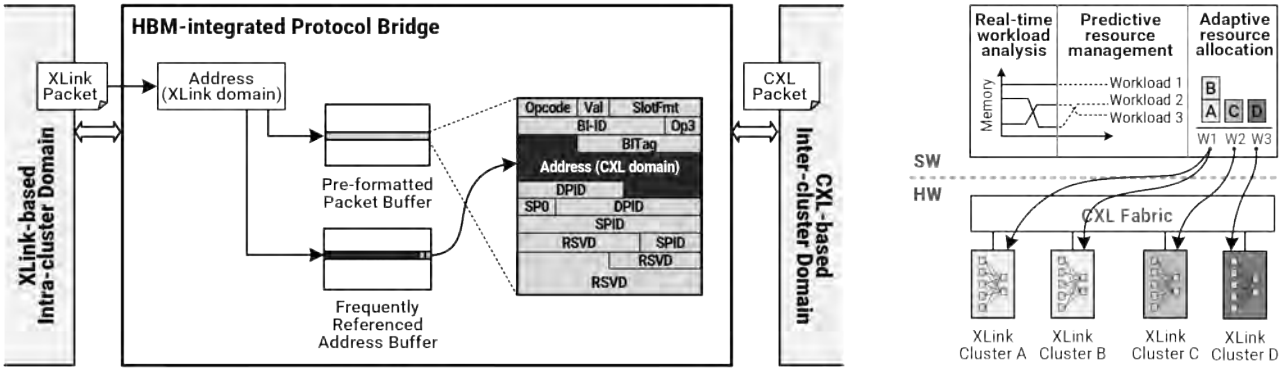
|그림 42| CXL-over-XLink에서의 캐시 일관성과 데이터 이동 관리.

집중할 수 있다. 병렬 연산에 최적화된 소프트웨어 커널이 연산을 수행하는 동안, 클러스터 간의 고속 메모리 데이터 전송은 CXL의 프로토콜 수준 캐시 일관성 정책이 하드웨어적으로 어떤 개입도 필요 없이 완벽히 처리해 준다. 특히 다양한 워크로드 실행에 있어서 접근 지역성을 보이는 데이터 접근은 가속기 내부의 캐시를 효과적으로 활용하여 성능과 연산 효율성을 극대화할 수 있다.

XLink와 CXL 통합 아키텍처를 위한 하드웨어 및 소프트웨어 최적화. 슈퍼클러스터 아키텍처에서 XLink와 CXL을 명확히 구분하여 사용하는 것은 설계상의 큰 이점을 제공한다. 그러나 서로 다른 두 기술을 실제로 통합할 때는 여러 구현상의 어려움이 존재할 수 있다. 특히, XLink 기반 클러스터 내부와 CXL 기반 클러스터 간 패브릭 도메인 사이에서 데이터 포맷과 프로토콜 변환 과정이 필요하게 되며, 이는 추가적인 지연을 유발할 수 있다. 이러한 오버헤드는 전반적인 성능 저하로 이어질 수 있으며, 특히 지연에 민감한 워크로드에 부정적인 영향을 줄 수 있다. 또한 클러스터 내부에 가속기 밀도가 높아지면서 냉각 요구가 증가하고, 일관성 메모리 트랜잭션 처리 시 인터커넥트의 신뢰성과 오류 관리에 대한 문제도 추가로 발생할 수 있다.

이러한 문제를 해결하기 위해서는 맞춤형 하드웨어 최적화가 필요할 수 있다. 그림 43a에서 보는 바와 같이, 특수 설계된 SoC 기반 브리지 인터페이스를 통해 프로토콜 간 데이터 포맷 변환을 신속하게 처리하도록 최적화하고, 간소화된 프로토콜을 사용하여 인터페이스 간 핸드쉐이크 오버헤드를 최소화할 수 있도록 간소화하여 구현할 수 있다. 특히, 브리지 인터페이스 내부에 고대역폭 메모리(HBM)를 통합하여, 프로토콜 변환 과정에서의 성능 저하를 더욱 완화할 수 있다. 예를 들어, 자주 접근되는 메모리 주소나 요청 결과를 미리 HBM에 저장하여 동일한 데이터 요청 시 다시 변환하는 과정 없이 즉시 사용할 수 있도록 하면, 앞서 언급된 지연을 크게 줄일 수 있다. 또한 CXL-over-XLink 기반 슈퍼클러스터 내부에 데이터 배치 전략을 지능적으로 설계하여 불필요한 데이터 이동을 최소화하면 전체 시스템의 성능을 더욱 향상시킬 수 있다.

이러한 하드웨어적 최적화 설계 방법 이외에도, 고급 오케스트레이션이나 소프트웨어 전략 등을 활용하여 성능을 개선시키는 것도 하나의 중요 포인트이다. 그림 43b에서처럼, 오케스트레이션 소프트웨어 프레임워크 등을 통해



(a) HBM이 통합된 특수 브리지 인터페이스. **(b)** 오케스트레이션 소프트웨어.

|그림 43| XLink와 CXL 통합 아키텍처를 위한 하드웨어 및 소프트웨어 최적화.

실시간 워크로드 모니터링과 예측 기반의 자원 관리, 적응형 자원 할당이 가능하며, 이는 CXL-over-XLink 기반의 슈퍼클러스터와 같은 대규모 통합 아키텍처의 운영 효율성 및 성능을 크게 높일 수 있다. 예를 들어, 특정 클러스터에서 외부로 접근하는 메모리 요청이 빈번히 관측되면, 소프트웨어는 데이터를 요청하는 클러스터로 해당 데이터를 물리적으로 이동시킴으로써 클러스터 내 가속기 간에 데이터 접근 및 처리가 일어나도록 최적화할 수 있다. 또한 데이터 중복성, 복제, 체크포인트(Checkpoint) 등의 강력한 내고장성 메커니즘을 소프트웨어를 통해 도입하면 이러한 대규모 다중 가속기 시스템의 안정성과 신뢰성을 더욱 높일 수 있다. 이를 통해 구성 요소의 고장이나 예기치 않은 데이터 손상, 인터커넥트 장애 등의 위험을 효과적으로 관리할 수 있으며, 결과적으로 지속적이고 안정적인 운영과 일관된 시스템 성능을 유지할 수 있을 것이다.

6.3. XLink와 경량화된 CXL 링크를 활용한 계층적 메모리 구성

현대 AI 워크로드의 다양한 메모리 성능 요구를 효과적으로 충족하기 위해, 본 서브섹션에서는 CXL-over-XLink 기반 슈퍼클러스터에 계층적 메모리 구성을 추가할 수 있도록 한 단계 발전시킨 확장형 구조를 제안한다. 이 구조에서 메모리 구성은 크게 두 가지 메모리 계층으로 나뉜다. 첫 번째는 XLink와 일관성 중심의 CXL(Coherence-Centric CXL)로 관리되는 고성능 로컬 메모리이며, 두 번째는 용량 중심의 CXL(Capacity-Oriented CXL)을 통한 확장 가능한 메모리 풀이다. 이 두 가지 다른 메모리 계층을 만들기 위해서 CXL을 경량화된 형태로 구현하는 것을 제시하며, 본 절 마지막에는 메모리 계층을 가지고 있는 슈퍼클러스터를 전략적으로 활용하기 위한 데이터 배치 및 관리 기법을 함께 제안하며 마무리한다.

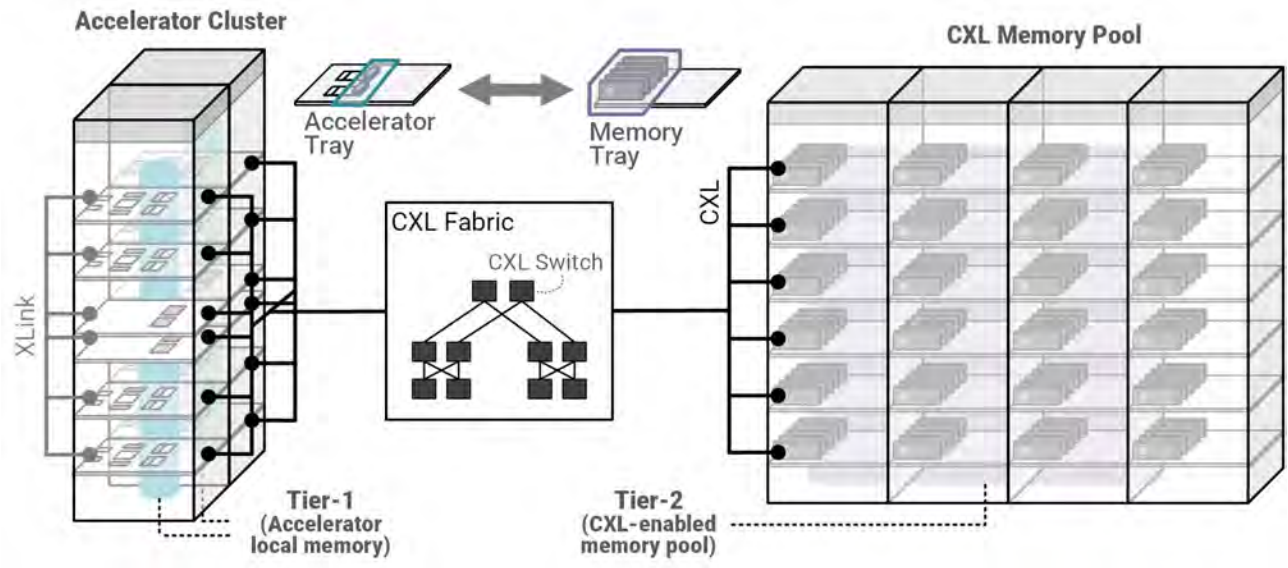
고성능 가속기 로컬 메모리: 일관성을 지원하는 XLink. 기존 CXL-over-XLink의 슈퍼클러스터 구조 내에서 개별 가속기 클러스터는 XLink로 연결되며, 각각의 가속기는 HBM 또는 고대역폭 특화 DDR 메모리와 같은 커스텀 메모리 기술을 사용한다. 이러한 가속기 구조들은 각각 이미 정해진 상태로 클러스터를 구조화하기 때문에 버전이나 용량이 다른 HBM들이 하나의 슈퍼클러스터 내에 있을 수 있으며, 또한 다른 유형의 이종 간 고속 메모리 등도 클러스터 내

혼재할 수 있다. 인프라의 스케일이 대규모가 됨에 따라, 실제 인프라 내에서 실행되는 워크로드와 모델에 따라서 각 클러스터 간 필요한 메모리 용량과 종류 그리고 접근 패턴 등이 모두 다를 수 있다. 슈퍼클러스터는 CXL-over-XLink에서 클러스터 간을 연결해 주는 CXL 패브릭이 이미 포함되어 있으므로 이를 잘 활용하면 클러스터 간에 흩어져 있는 고성능 메모리를 논리적으로 통합하여 일관된 가속기 로컬 메모리 계층을 형성할 수 있다.

우선 클러스터 내부의 로컬 메모리는 XLink를 사용하여 통합된 메모리 주소 공간을 만들어낼 수 있다. 다양한 방법으로 주소 공간을 생성할 수도 있겠지만 XLink의 프로토콜 명세와 동작 방식을 생각해 볼 때 가장 기본적인 방식은 각 가속기의 메모리 모듈마다 정적으로 파티션된 블럭으로 인식하여 이를 연속적인 주소 공간으로 할당해 통합된 메모리 주소 공간을 만드는 것이다. 예를 들어 UALink는 각 가속기의 메모리를 정적으로 나누어 NUMA 형태의 통합된 메모리 도메인을 만들 수 있으며, NVLink는 가상화를 통해 장치 간 통합된 주소 공간을 구성한다. 그러나 이런 방식으로 통합된 메모리는 각각 정적 메모리 영역을 넘어가는 공간에 대한 공유가 일어날 수가 없어서 소프트웨어나 펌웨어가 개입하여 데이터를 복사해 줘야 하며 프로토콜 상의 캐시 일관성이 없으므로 데이터 자체도 공유할 수 없다. 따라서 가속기가 로컬로 소유하지 않은 메모리 영역에 접근할 때는 XLink를 통한 데이터 전송이 필요하며, 이는 추가적인 지연을 유발한다. 특히 서로 다른 클러스터 사이의 메모리 접근이 발생하면 이러한 문제는 더욱 두드러져 성능 저하로 이어진다.

이러한 성능 저하 및 지연 문제를 해결하기 위해 일관성 중심의 CXL을 활용할 수 있다. CXL을 수정하지 않고 CXL-over-XLink의 클러스터 간 CXL 연결만을 그대로 사용한다는 가정 하에, 각 클러스터는 메모리 주소 공간 중 일부를 지정하고 해당 주소 공간에 대해서만 클러스터 간 CXL 패브릭에서 캐시 일관성을 유지해주는 방식으로 특정한 응용과 데이터에 대해서 공유하거나 일관성을 확보해 줄 수 있다. 다시 말해, 가속기 로컬 메모리 계층의 일부 영역에 대해 캐시 일관성을 유지하면서 데이터 공유를 가능하게 하고, 나머지 영역에서는 XLink를 통한 데이터 복사 및 이동을 통해서 통합된 주소 공간을 관리하게 하는 것이다. 이러한 클러스터 간 CXL 패브릭을 이용한 부분적 일관성 허용 방법은 각각의 AI 워크로드들 실행에 있어서 클러스터 내 데이터 지역성을 높이고 성능을 개선할 수 있다. 또한 데이터 접근이 클러스터 내에서 일어나게 되고 지역성이 높아지면, 빈번하게 접근되는 데이터에 대해서는 애초에 가속기 온칩 캐시에서 자동으로 관리되어 데이터 이동 등을 원천적으로 해결할 수 있다.

좀 더 진보적인 방법으로는 데이터 공유와 캐시 일관성 요구가 높은 워크로드에서는 CXL을 일관성에 초점을 맞추고 경량화하여 클러스터 내부에서도 적극적으로 활용할 수 있도록 구성할 수 있다. 이 경우, 각 가속기에 전용의 CXL 컨트롤러 로직을 추가하여, XLink 컨트롤러와 가깝게 배치하거나 컨트롤러 내부에 직접 통합하여 설계하고 구현할 수 있다. 이러한 구성에서 가장 큰 이점은 일관성 중심의 경량화된 CXL이 XLink와 함께 클러스터 내에도 존재하게 되는 것으로 모든 GPU나 가속기가 하나의 메모리 공간을 보고 데이터를 모두 공유하는 구조를 만들어낼 수 있다. 집합 연산이나 명시적 데이터 이동이 완전히 사라지게 만들 수 있으므로 상당한 성능 이득을 볼 수 있다. 또한, 비록 이러한 구조는 SoC 구현 복잡성, 비용, 중복된 데이터 전송 등의 문제를 야기할 가능성이 있지만, CXL 프로토콜의 불필요한 기능들을 제거하고 캐시 일관성에 초점을 맞추는 최적화를 통해 효과적으로 완화할 수 있다. 이러한 XLink와 CXL의 통합 컨트롤러는 여러 가지 방법으로 다양하게 구현될 수 있지만, 기본적으로 중복 기능들을 제거하기 위해 대용량 데이터 이동은 XLink를 통해 처리하고, 가속기의 컨트롤러는 최적화된 CXL.cache 하위 프로토콜만 구현하여 오직 일관성 트래픽만 관리하고 처리하는 방식을 채택할 수 있다. 이 접근법은 세부 컨트롤러 설계와 프로토콜 관리를 요구하지만, 결과적으로 캐시 일관성 향상, 데이터 관리 간소화, 그리고 슈퍼클러스터 전체의 계산 효율성 향상과 같은 성능적 이점을 제공한다.



|그림 44| CXL 메모리 풀로 구성된 계층적 메모리 구조.

이와 같이 XLink 기반의 로컬 메모리와 일관성 중심 CXL을 결합하면 가속기 노드 수준의 낮은 지연과 캐시 일관성 요구를 효과적으로 만족시킬 수 있다. 그러나 현대 AI 워크로드는 종종 랙 수준의 가속기 로컬 메모리 용량 총합을 초과하는 대규모 메모리 공간을 요구하기 때문에, 별도의 확장 가능한 용량 중심 메모리 풀이 추가로 필요하다. 이를 위해서는 다음에 제안할 컴포저블 메모리 풀을 사용하는 보완 전략이 필요하다.

용량 중심의 컴포저블 메모리 풀: CXL. 가속기 로컬 메모리는 성능이 중요한 데이터를 빠르게 처리하는 데 효과적이다. 그러나 많은 AI 워크로드는 성능을 일부 희생하더라도 훨씬 더 큰 용량의 메모리를 필요로 한다. 예를 들어 대규모 임베딩 테이블이나 캐시, 외부 지식 데이터베이스와 같은 경우가 이에 해당한다. 이를 해결하기 위해, 본 보고서에서는 슈퍼클러스터 아키텍처 내부에 별도의 컴포저블 메모리 풀을 구성하여 용량을 유연하게 확장할 수 있는 2계층 구조를 제안한다.

이 2계층의 컴포저블 메모리 풀은 주로 앞서 설명한 메모리 트레이로 구성되며, 그림 44에서 보이듯이 가속기 클러스터와 분리된 상태에서 CXL 패브릭을 통해 연결된다. CXL-over-XLink 기반의 슈퍼컴퓨터에서 컴포저블 메모리 풀의 특성은 이미 일관성 중심의 CXL과 XLink의 관리로 가속기 간 통합된 메모리 뷰를 1계층 메모리로 가지고 있다는 것이다. 따라서 2계층 메모리까지 데이터 접근이 생기는 경우는 최소한 랙 수준의 가속기 메모리 총량을 넘어가는 수준으로 과거 RAG와 같은 응용에서 데이터를 스토리지 또는 분산 파일 시스템으로부터 가져오는 경우에 가까울 것이다. 이러한 경우는 수 밀리초에서 수십 초까지 가속기로 데이터가 올라오는 시간이 생길 수 있는데 2계층의 컴포저블 메모리 풀은 패브릭의 스위치에 따라 다르지만, 여전히 수십에서 수백 나노초 미만으로 상당한 이득을 얻을 수 있다. 물론 여기서 가장 큰 역할은 메모리 용량이며, 컴포저블 메모리 풀의 최대 용량 효율을 달성하기 위해, 메모리 풀을 구성하는 메모리 트레이 각각에는 CPU나 가속기와 같은 연산 장치를 배치하지 않고 순수한 메모리 자원만을 집약한다.

메모리 트레이의 물리적 배치는 스위치 한 홉의 수와 지연과 관계가 있는 것으로, CXL 패브릭이 닿을 수 있는 곳이면 CXL-over-XLink 기반 슈퍼클러스터 내 어디든 관계가 없다. 또한 데이터센터 디자이너의 공간 관리에 대한

요구사항에 맞춰 다양한 방식으로 메모리 트레이를 구성하되 2계층의 메모리 풀에 대한 주소 공간을 패브릭이 알 수 있게 하여 배치하거나 가상적 관리 기법을 통해 논리적으로 연결할 수 있다. 물론 메모리 트레이를 가속기 클러스터 가까이에 배치하면, 성능이 떨어지는 스케일아웃 데이터 접근 방식에 대한 의존을 크게 줄일 수 있다. 실제 운영 환경에서 이와 같은 외부 메모리 자원은 계층적 메모리 아키텍처의 2계층을 형성하며, 용량 확장에 특화된 역할을 수행한다.

일관성 중심의 CXL 접근 방법에서도 그랬지만, 현재 용량 중심의 CXL 또한 기존 CXL-over-XLink 패브릭을 그대로 쓸 수도 있고 이를 개선할 수하여 대규모 워크로드에 더욱 최적화 할 수도 있다. 다시 말해 이미 1계층 로컬 메모리가 캐시 일관성을 유지하며 지연에 민감한 데이터를 처리하므로, 2계층 메모리 풀은 오로지 메모리 용량에 초점을 맞추고 다른 기능은 단순화하고 좀 더 비용 효율적으로 만들 수 있다. 예를 들어, 모든 메모리 트레이 사이에서 캐시 일관성을 유지할 필요는 없다. 따라서 컨트롤러의 설계를 가볍게 하고 효율성을 최대화하기 위해 CXL.cache 또는 CXL.io 프로토콜을 스위치나 엔드포인트에서 비활성화하도록 설계하고 구성할 수 있다. 특히, 1계층 메모리가 배타적 캐시로서 충분한 경우, 2계층 컴포저블 메모리 풀은 CXL.mem을 생략하고 대량 데이터 전송을 위한 CXL.io 만으로 구성할 수도 있다. 다만 어떤 방식을 선택하든 가속기 로컬 메모리와 컴포저블 메모리 풀 사이에는 지속적인 데이터 전송이 발생하므로, 충분한 수의 CXL 패브릭 포트를 제공하여 데이터 이동 성능을 최적화하는 것이 중요하다.

프로토콜의 단순화 외에도, 2계층 용량 중심 메모리 풀을 위한 추가 최적화가 가능하다. 5.2절에서 논의한 것처럼, 메모리 트레이는 고속 DRAM 대신 저속 DRAM 인터페이스(예를 들어 LPDDR이나 DDR3)를 활용하여 비용을 절감할 수 있다. 또는, 대용량 플래시 메모리와 소량의 고속 HBM을 결합한 하이브리드 메모리 트레이를 구성하면, 전체 용량을 극대화하면서도 1계층 가속기 로컬 메모리와의 데이터 전송에 필요한 성능을 제공할 수 있다. 또한, 여러 층이나 건물을 가로지르는 긴 물리적 거리에서도 안정적인 데이터 전송과 버퍼링을 지원하기 위해, 슈퍼클러스터의 계층적 메모리 구조에서는 PCIe PHY 대신 실리콘 포토닉스(Silicon Photonics)와 같은 광학 기술을 적용하여 CXL 연결 성능을 더욱 높일 수 있다.

계층적 데이터 배치 및 관리 전략. 앞서 이야기된 것처럼 가속기 로컬 메모리(XLink 기반)와 컴포저블 메모리 풀(CXL 기반)을 결합하면, 성능과 메모리 용량을 모두 크게 확장할 수 있는 새로운 기회를 제공할 수 있다. 하지만 이러한 계층적 메모리 구조를 최대한 활용하기 위해서는 데이터를 효율적으로 배치하고, 자원을 지능적으로 관리하는 소프트웨어 프레임워크의 도움이 필요할 수도 있다. 특히, 가속기 클러스터 간 계산 작업의 동적 배분과 CXL 메모리 풀의 효율적 할당을 동시에 지원하는 통합 관리 방식은 AI 데이터센터의 자원 활용도를 높이고, 성능과 운영 효율성을 균형 있게 유지하는 데 큰 도움이 될 수 있다.

다시 말해 계층적 메모리 아키텍처의 효과를 극대화하려면 각 계층의 성능 특성을 고려한 데이터 배치 전략이 중요한데, 이런 부분은 하드웨어가 직접 들어가서 전략을 수행하는 것보다는 소프트웨어가 진행하는 것이 유리하다. 구체적으로 데이터 배치 결정 시에는 데이터의 접근 빈도와 지연 민감도를 명확하게 평가해야 하는데, 이런 부가적인 부분은 그 기능 자체가 하드웨어로 구성될 필요가 없으므로 소프트웨어로 구성하고 전략을 정교하게 만드는 것이 좋을 것이다. 예를 들어, 활성화 상태, 임베딩 벡터, 어텐션 캐시 등 빈번히 접근되고 빠른 응답이 필요한 데이터는 가속기 로컬 메모리에 저장해야 하는 반면, 용량이 크고 지연에 상대적으로 덜 민감한 데이터는 용량 중심의 CXL 컴포저블 메모리 풀에 배치하는 것이 적합하다. 이러한 계층적 접근법을 구현하기 위해, 다양한 런타임 정보들을 모니터링하고 각 데이터의 특성에 최적화된 형태로 배치하여 자원 활용을 최대화하고 성능을 높일 수 있다.

좀 더 나아가, 정교한 데이터 배치 전략의 효율적인 운영을 위해서는 소프트웨어 기반의 고급 오케스트레이션 프레임워크를 추가하여 관리할 수도 있다. 예를 들어, 워크로드 패턴과 데이터 접근 빈도의 변화를 예측하여 데이터 위치를 동적으로 조정하는 예측 기반 데이터 마이그레이션 알고리즘을 도입할 수 있다. 또한, 접근 빈도에 따라 우선순위를 부여하는 온도 인지 캐싱 정책과 기계학습 기반의 지능적 프리페칭을 통해 데이터 이동의 오버헤드와 지연을 최소화할 수 있다. 그러나 데이터 마이그레이션이 지나치게 빈번해지면 성능 저하가 발생할 수 있으므로, 하드웨어 최적화, XLink와 CXL 간 프로토콜 변환 인터페이스의 효율적 설계, 신중한 데이터 이동 정책 등 포괄적 접근이 필요하다.

전체적으로 다시 한번 정리해 보면, CXL과 XLink 등을 통해 구성되는 대규모 다중 가속기 시스템에서 대규모 AI 응용 프로그램은 계층적 데이터 관리 접근법에서 큰 혜택을 얻을 수 있다. 예컨대, 실시간 추론 작업은 가속기 로컬 메모리를 활용해 빠른 데이터 처리를 수행할 수 있으며, 임베딩 조회나 외부 데이터 검색과 같은 대규모 작업은 CXL 컴포저블 메모리 풀을 통해 효율적으로 처리할 수 있다. 따라서 각 메모리 계층의 역할을 명확히 구분하고 전략적으로 데이터를 배치하면, 현대 AI 데이터센터의 다양한 요구사항을 효과적으로 만족할 수 있는 성능과 자원 활용 최적화를 실현할 수 있다.

7. 결론

본 기술 보고서에서는 현대 AI 워크로드를 확장하는 데 있어 GPU 중심 아키텍처의 한계를 분석하였다. 특히 메모리 용량, 장치 간 통신, 자원 관리와 관련된 문제점들이 주요한 성능 병목으로 지적되었다. 이를 극복하기 위해 메모리와 연산 자원을 동적으로 분리 및 할당할 수 있는 CXL 기반의 모듈형 데이터센터 아키텍처를 제안하였다.

이러한 아키텍처의 실증적 평가를 위해 RAG, Graph-RAG, DLRM, MPI 기반의 다양한 AI 워크로드를 활용하였으며, 기존 RDMA 기반 시스템 대비 지연, 통신 오버헤드, 메모리 관리 복잡성이 현저히 개선됨을 확인하였다. 이 결과를 통해, CXL의 캐시 일관성 메모리 공유와 자원 컴포저빌리티 기능이 자원 효율성을 극대화하고 운영 유연성을 높이는 데 효과적임을 입증하였다.

추가적으로 UALink와 NVLink 같은 전용 가속기 중심 인터커넥트 기술을 CXL과 결합한 하이브리드 아키텍처를 분석하였다. 이 결합을 통해 가속기 간의 효율적인 통신을 유지하면서도 메모리 확장성과 데이터 공유의 유연성을 확보할 수 있었다.

마지막으로, 실제 데이터센터에서 컴포저블 CXL 인프라를 구축할 때의 주요 아키텍처적 고려 사항을 제시하였다. 구체적으로, 전용 메모리 풀링, 적응형 데이터 배치, 가속기 중심 자원 관리, 고급 중앙집중식 모니터링의 중요성을 논의하였다. 향후 연구는 실질적인 배포 환경의 문제를 해결하고, 고급 오케스트레이션 기법을 개발하며, 하이브리드 인터커넥트 전략을 더욱 세밀히 최적화하는 데 집중되어야 한다. 이러한 지속적 노력을 통해 점점 더 높은 성능을 요구하는 현대 AI 워크로드에 대한 컴포저블 CXL 인프라의 잠재력을 완전히 실현할 수 있을 것이다.

8. 참고문헌

[1] J. Schmidhuber, "Annotated history of modern ai and deep learning," arXiv preprint arXiv:2212.11279, 2022.

[2] A. Toosi, A. G. Bottino, B. Saboury, E. Siegel, and A. Rahmim, "A brief history of ai: how to prevent another winter," PET clinics, 2021.

[3] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science, 2015.

[4] J. Hendler, "Avoiding another ai winter," IEEE Intelligent Systems, 2008.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, 2012.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (NAACL-HLT'19), 2019.

[7] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan et al., "The llama 3 herd of models," arXiv preprint arXiv:2407.21783, 2024.

[8] W. Chu, X. Xie, J. Yu, J. Wang, A. Phanishayee, C. Tang, Y. Hao, J. Huang, M. Ozdal, J. Wang et al., "Scaling llama 3 training with efficient parallelism strategies," in Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA'25), 2025.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in neural information processing systems, 2014.

[10] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," Advances in neural information processing systems, 2014.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE transactions on pattern analysis and machine intelligence, 2013.

[12] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan, "Intrinsic dimension of data representations in deep neural networks," Advances in Neural Information Processing Systems, 2019.

[13] C. Ruiz, H. Ren, K. Huang, and J. Leskovec, "High dimensional, tabular deep learning with an auxiliary knowledge graph," Advances in Neural Information Processing Systems, 2023.

[14] C. Hawthorne, A. Jaegle, C. Cangea, S. Borgeaud, C. Nash, M. Malinowski, S. Dieleman, O. Vinyals, M. Botvinick, I. Simon et al., "General-purpose, long-context autoregressive modeling with perceiver ar," in International Conference on Machine Learning (ICML'22), 2022.

[15] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," Advances in neural information processing systems, 2016.

[16] K. Chandra, A. Xie, J. Ragan-Kelley, and E. Meijer, "Gradient descent: The ultimate optimizer," Advances in Neural Information Processing Systems, 2022.

[17] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016.

[18] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," SIAM review, 2018.

[19] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in Neural networks: Tricks of the trade: Second edition, 2012.

[20] NVIDIA, "Nvidia blackwell architecture technical brief: Built for the age of ai reasoning," https://resources.nvidia.com/en-us-blackwell-architecture, 2024.

[21] ——, “Nvidia blackwell datasheet: The engine of the new industrial revolution,” https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/, 2025.

[22] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, “A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives,” Journal of Systems Architecture, 2022.

[23] M. Pandey, M. Fernandez, F. Gentile, O. Isayev, A. Tropsha, A. C. Stern, and A. Cherkasov, “The transformational role of gpu computing and deep learning in drug discovery,” Nature Machine Intelligence, 2022.

[24] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, “Zero: Memory optimizations toward training trillion parameter models,” in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC’20), 2020.

[25] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, “Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters,” in Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (KDD’20), 2020.

[26] S. Sano, Y. Bando, K. Hiwada, H. Kajihara, T. Suzuki, Y. Nakanishi, D. Taki, A. Kaneko, and T. Shiozawa, “Gpu graph processing on cxl-based microsecond-latency external memory,” in Proceedings of the SC’23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W’23), 2023.

[27] S. W. Min, V. S. Mailthody, Z. Qureshi, J. Xiong, E. Ebrahimi, and W.-m. Hwu, “Emogi: Efficient memory-access for out-of-memory graph-traversal in gpus,” arXiv preprint arXiv:2006.06890, 2020.

[28] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro et al., “Efficient large-scale language model training on gpu clusters using megatron-lm,” in Proceedings of the international conference for high performance computing, networking, storage and analysis (SC’21), 2021.

[29] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with pagedattention,” in Proceedings of the 29th symposium on operating systems principles (SOSP’23), 2023.

[30] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett et al., “H2o: Heavy-hitter oracle for efficient generative inference of large language models,” Advances in Neural Information Processing Systems, 2023.

[31] M. Adnan, A. Arunkumar, G. Jain, P. J. Nair, I. Soloveychik, and P. Kamath, "Keyformer: Kv cache reduction through key tokens selection for efficient generative inference," Proceedings of Machine Learning and Systems, 2024.

[32] B. Li, X. Wang, J. Wang, Y. Liu, Y. Gong, H. Lu, W. Dang, W. Zhang, X. Huang, M. Chen et al., "Tccl: Co-optimizing collective communication and traffic routing for gpu-centric clusters," in Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing (SIGCOMM'24), 2024.

[33] Y. Jin, C.-F. Wu, D. Brooks, and G.-Y. Wei, "$s3$: Increasing gpu utilization during generative inference for higher throughput," Advances in Neural Information Processing Systems, 2023.

[34] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'21), 2021.

[35] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and E. P. Xing, "Poseidon: An efficient communication architecture for distributed deep learning on gpu clusters," in 2017 USENIX Annual Technical Conference (USENIX ATC 17), 2017.

[36] F. V. Zacarias, K. Palli, S. Vazhkudai, and E. Grevelink, "A memory perspective: The effects of fine-tuning llms with high-bandwidth memory," 2024.

[37] L. Fusco, M. Khalilov, M. Chrapek, G. Chukkapalli, T. Schulthess, and T. Hoefler, "Understanding data movement in tightly coupled heterogeneous systems: A case study with the grace hopper superchip," arXiv preprint arXiv:2408.11556, 2024.

[38] H. Miao, M. Jeon, G. Pekhimenko, K. S. McKinley, and F. X. Lin, "Streambox-hbm: Stream analytics on high bandwidth hybrid memory," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'19), 2019.

[39] M. Zhu, Y. Zhuo, C. Wang, W. Chen, and Y. Xie, "Performance evaluation and optimization of hbm-enabled gpu for data-intensive applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018.

[40] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," Journal of Machine Learning Research, 2022.

[41] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in International conference on machine learning (ICML'20), 2020.

[42] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark et al., "Improving language models by retrieving from trillions of tokens," in International conference on machine learning (ICML'22), 2022.

[43] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," arXiv preprint arXiv:1909.08053, 2019.

[44] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" arXiv preprint arXiv:2002.08910, 2020.

[45] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," in Proceedings of the 27th ACM symposium on operating systems principles (SOSP'19), 2019.

[46] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat et al., "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[47] W. Li, X. Liu, Y. Li, Y. Jin, H. Tian, Z. Zhong, G. Liu, Y. Zhang, and K. Chen, "Understanding communication characteristics of distributed training," in Proceedings of the 8th Asia-Pacific Workshop on Networking (APNet'24), 2024.

[48] S. Hsia, A. Golden, B. Acun, N. Ardalani, Z. DeVito, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Mad-max beyond single-node: Enabling large machine learning model acceleration on distributed systems," in 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA'24), 2024.

[49] L.-W. Chang, W. Bao, Q. Hou, C. Jiang, N. Zheng, Y. Zhong, X. Zhang, Z. Song, C. Yao, Z. Jiang et al., "Flux: Fast software-based communication overlap on gpus through kernel fusion," arXiv preprint arXiv:2406.06858, 2024.

[50] C. Jiang, Y. Tian, Z. Jia, S. Zheng, C. Wu, and Y. Wang, "Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping," Proceedings of Machine Learning and Systems, 2024.

[51] G. Huang, H. Li, L. Qin, J. Huang, Y. Kang, Y. Ding, and Y. Xie, "Traci: Network acceleration of input-dynamic communication for large-scale deep learning recommendation model," in Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA'25), 2025.

[52] W. Hou, J. Zhang, Z. Wang, and M. Liu, “Understanding routable pcie performance for composable infrastructures,” in 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI’24), 2024.

[53] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, “Understanding pcie performance for end host networking,” in Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’18), 2018.

[54] CXL Consortium, “Compute express link (cxl) specification revision 1.0,” https://computeexpresslink.org/wp-content/uploads/2024/02/CXL-1.0-Specification.pdf, 2019.

[55] ——, “Compute express link (cxl) specification revision 2.0,” https://computeexpresslink.org/wp-content/uploads/2024/02/CXL-2.0-Specification.pdf, 2020.

[56] ——, “Compute express link (cxl) specification revision 3.2 version 1.0,” https://computeexpresslink.org/cxl-specification/, 2024.

[57] S.-P. Yang, M. Kim, S. Nam, J. Park, J.-Y. Choi, E. H. Nam, E. Lee, S. Lee, and B. S. Kim, “Overcoming the memory wall with cxl-enabled ssd,” in 2023 USENIX Annual Technical Conference (USENIX ATC 23), 2023.

[58] D. Das Sharma, R. Blankenship, and D. Berger, “An introduction to the compute express link (cxl) interconnect,” ACM Computing Surveys, 2024.

[59] M. Jung, “Hello bytes, bye blocks: Pcie storage meets compute express link for memory expansion (cxl-ssd),” in Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage’22), 2022.

[60] H. Li, D. S. Berger, L. Hsu, D. Ernst, P. Zardoshti, S. Novakovic, M. Shah, S. Rajadnya, S. Lee, I. Agarwal et al., “Pond: Cxl-based memory pooling systems for cloud platforms,” in Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’23), 2023.

[61] D. Gouk, S. Lee, M. Kwon, and M. Jung, “Direct access, high-performance memory disaggregation with directcxl,” in 2022 USENIX Annual Technical Conference (USENIX ATC 22), 2022.

[62] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al., “Retrieval-augmented generation for knowledge-intensive nlp tasks,” Advances in neural information processing systems (NIPS’20), 2020.

[63] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, “Active retrieval augmented generation,” in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP’23), 2023.

[64] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'24), 2024.

[65] T. Zhang, J. Yi, Z. Xu, and A. Shrivastava, "Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization," Advances in Neural Information Processing Systems, 2024.

[66] W. Lee, J. Lee, J. Seo, and J. Sim, "Infinigen: Efficient generative inference of large language models with dynamic kv cache management," in 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24), 2024.

[67] Y. Liu, H. Li, Y. Cheng, S. Ray, Y. Huang, Q. Zhang, K. Du, J. Yao, S. Lu, G. Ananthanarayanan et al., "Cachegen: Kv cache compression and streaming for fast large language model serving," in Proceedings of the ACM SIGCOMM 2024 Conference (SIGCOMM'24), 2024.

[68] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava, "Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time," Advances in Neural Information Processing Systems, 2023.

[69] UALink Consortium, "Ualink 200g revision 1.0 specification," https://ualinkconsortium.org/specification/, 2025.

[70] N. Kalyanasundharam, "Introducing ualink 200g 1.0 specification," https://ualinkconsortium.org/wp-content/uploads/2025/04/UALink-1.0-White_Paper_FINAL.pdf, 2025.

[71] A. Ishii and R. wells, "The nvlink-network switch: Nvidia's switch chip for high communication-bandwidth superpods," in Hot Chips 34 Symposium (HCS'34), 2022.

[72] R. Merritt, "What is nvlink?" https://blogs.nvidia.com/blog/what-is-nvidia-nvlink, 2023.

[73] NVIDIA, "Nvidia nvlink and nvlink switch: The building blocks of high-speed, multi-gpu communication for feeding large datasets faster into models and rapidly exchanging data between gpus," https://www.nvidia.com/en-us/data-center/nvlink/, 2025.

[74] A. Sharpiro, "Nvidia unveils nvlink fusion for industry to build semi-custom ai infrastructure with nvidia partner ecosystem," https://nvidianews.nvidia.com/news/nvidia-nvlink-fusion-semi-custom-ai-infrastructure-partner-ecosystem, 2025.

[75] NVIDIA, "Nvidia nvlink fusion: Semi-custom ai infrastructure with industry-proven ai scale-up performance," https://www.nvidia.com/en-us/data-center/nvlink-fusion/, 2025.

[76] A. Kalia, M. Kaminsky, and D. G. Andersen, “Design guidelines for high performance rdma systems,” in 2016 USENIX annual technical conference (USENIX ATC 16), 2016.

[77] J. Wang, B. Lin, J. Zhang, M. Sun, and Y. Pan, “An optimized rdma qp communication mechanism for hyperscale ai infrastructure,” Cluster Computing, 2025.

[78] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, “Rdma over commodity ethernet at scale,” in Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM’16), 2016.

[79] A. Gangidi, R. Miao, S. Zheng, S. J. Bondu, G. Goes, H. Morsy, R. Puri, M. Riftadi, A. J. Shetty, J. Yang et al., “Rdma over ethernet for distributed training at meta scale,” in Proceedings of the ACM SIGCOMM 2024 Conference (SIGCOMMM’24), 2024.

[80] K. Ueno and T. Suzumura, “Highly scalable graph search for the graph500 benchmark,” in Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing (HPDC’12), 2012.

[81] NASA, “Nasa parallel benchmarks,” https://www.nas.nasa.gov/software/npb.html, 2024.

[82] T. Pohl, “Lattice boltzmann method (lbm) benchmarks description,” https://www.spec.org/cpu2017/Docs/benchmarks/619.lbm_s.html, 2020.

[83] S. McIntosh-Smith, M. Martineau, T. Deakin, G. Pawelczak, W. Gaudin, P. Garrett, W. Liu, R. Smedley-Stevenson, and D. Beckingsale, “Tealeaf: A mini-application to enable design-space explorations for iterative sparse linear solvers,” in 2017 IEEE International Conference on Cluster Computing (CLUSTER’17), 2017.

[84] Ł. Łach and D. Svyetlichnyy, “Advances in numerical modeling for heat transfer and thermal management: a review of computational approaches and environmental impacts,” Energies, 2025.

[85] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” Advances in neural information processing systems, 2014.

[86] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina et al., “State-of-the-art speech recognition with sequence-to-sequence models,” in 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP’18), 2018.

[87] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, “Multi-task sequence to sequence learning,” arXiv preprint arXiv:1511.06114, 2015.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, 2017.

[89] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., "Improving language understanding by generative pre-training," 2018.

[90] OpenAI, "New models and developer products announced at devday," https://openai.com/index/new-models-and-developer-products-announced-at-devday/, 2024.

[91] Meta, "Llama 4: Leading intelligence. unrivaled speed and efficiency," https://www.llama.com/, 2025.

[92] OpenAI, "Gpt-4 turbo," https://platform.openai.com/docs/models/gpt-4-turbo, 2025.

[93] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen et al., "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," arXiv preprint arXiv:2507.06261, 2025.

[94] A. Thompson, "Report: Google DeepMind Gemini," https://lifearchitect.ai/gemini-report/, 2024.

[95] A. Graves, "Long short-term memory," Supervised sequence labelling with recurrent neural networks, 2012.

[96] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[97] A. Graves, "Generating sequences with recurrent neural networks," arXiv preprint arXiv:1308.0850, 2013.

[98] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," Advances in neural information processing systems, 2020.

[99] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young et al., "Scaling language models: Methods, analysis & insights from training gopher," arXiv preprint arXiv:2112.11446, 2021.

[100] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann et al., "Palm: Scaling language modeling with pathways," Journal of Machine Learning Research, 2023.

[101] B. Workshop, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon et al., "Bloom: A 176b-parameter open-access multilingual language model," arXiv preprint arXiv:2211.05100, 2022.

[102] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, 1986.

[103] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," arXiv preprint arXiv:2105.08919, 2021.

[104] S. Kato and K. Hotta, "Mse loss with outlying label for imbalanced classification," arXiv preprint arXiv:2107.02393, 2021.

[105] J. Ren, M. Zhang, C. Yu, and Z. Liu, "Balanced mse for imbalanced visual regression," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR'22), 2022.

[106] S. Kullback and R. A. Leibler, "On information and sufficiency," The annals of mathematical statistics, 1951.

[107] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," Advances in neural information processing systems, 2018.

[108] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," IEEE access, 2019.

[109] E. Gordon-Rodriguez, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham, "Uses and abuses of the cross-entropy loss: Case studies in modern deep learning," 2020.

[110] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in International conference on Machine learning (ICML'23), 2023.

[111] R. Fletcher and M. J. Powell, "A rapidly convergent descent method for minimization," The computer journal, 1963.

[112] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 2002.

[113] W. A. Gardner, "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique," Signal processing, 1984.

[114] S.-i. Amari, "Backpropagation and stochastic gradient descent method," Neurocomputing, 1993.

[115] L. Eon Bottou, “Online learning and stochastic approximations,” Online learning in neural networks, 1998.

[116] M. Hardt, B. Recht, and Y. Singer, “Train faster, generalize better: Stability of stochastic gradient descent,” in International conference on machine learning (ICML’16), 2016.

[117] X. Li and F. Orabona, “On the convergence of stochastic gradient descent with adaptive stepsizes,” in The 22nd international conference on artificial intelligence and statistics (AISTATS’19), 2019.

[118] Y. Tian, Y. Zhang, and H. Zhang, “Recent advances in stochastic gradient descent in deep learning,” Mathematics, 2023.

[119] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[120] Z. Zhang, “Improved adam optimizer for deep neural networks,” in 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS’18), 2018.

[121] S. Bock, J. Goppold, and M. Weiß, “An improvement of the convergence proof of the adam-optimizer,” arXiv preprint arXiv:1804.10587, 2018.

[122] S. Mehta, C. Paunwala, and B. Vaidya, “Cnn based traffic sign classification using adam optimizer,” in 2019 international conference on intelligent computing and control systems (ICCS’19), 2019.

[123] D. Yi, J. Ahn, and S. Ji, “An effective optimization method for machine learning based on adam,” Applied Sciences, 2020.

[124] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th international conference on machine learning (ICML’10), 2010.

[125] M. M. Lau and K. H. Lim, “Review of adaptive activation function in deep neural network,” in 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES’18), 2018.

[126] B. Neyshabur, Y. Wu, R. R. Salakhutdinov, and N. Srebro, “Path-normalized optimization of recurrent neural networks with relu activations,” Advances in Neural Information Processing Systems, 2016.

[127] F. Godin, J. Degrave, J. Dambre, and W. De Neve, “Dual rectified linear units (drelus): A replacement for tanh activation functions in quasi-recurrent neural networks,” Pattern Recognition Letters, 2018.

[128] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” Neurocomputing, 2022.

[129] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” arXiv preprint arXiv:1409.1259, 2014.

[130] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 1998.

[131] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” arXiv preprint arXiv:1606.01781, 2016.

[132] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” arXiv preprint arXiv:1508.04025, 2015.

[133] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[134] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, “A high-speed and low-complexity architecture for softmax function in deep learning,” in 2018 IEEE asia pacific conference on circuits and systems (APCCAS’18), 2018.

[135] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks,” arXiv preprint arXiv:1612.02295, 2016.

[136] H. Peng, J. Li, Y. Song, and Y. Liu, “Incrementally learning the hierarchical softmax function for neural language models,” in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI’17), 2017.

[137] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” arXiv preprint arXiv:1703.03130, 2017.

[138] S. Serrano and N. A. Smith, “Is attention interpretable?” arXiv preprint arXiv:1906.03731, 2019.

[139] C. A. Córdova Sáenz and K. Becker, “Assessing the use of attention weights to interpret bert-based stance classification,” in IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT’21), 2021.

[140] S. Wiegreffe and Y. Pinter, “Attention is not not explanation,” arXiv preprint arXiv:1908.04626, 2019.

[141] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” arXiv preprint arXiv:1803.02155, 2018.

[142] H. Zhao, J. Jia, and V. Koltun, “Exploring self-attention for image recognition,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR’20), 2020.

[143] F. X. Gibbons, “Self-attention and behavior: A review and theoretical update,” Advances in experimental social psychology, 1990.

[144] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in International conference on machine learning (ICML’19), 2019.

[145] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, “Stand-alone self-attention in vision models,” Advances in neural information processing systems, 2019.

[146] N. Kitaev, Ł. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” arXiv preprint arXiv:2001.04451, 2020.

[147] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang et al., “Big bird: Transformers for longer sequences,” Advances in neural information processing systems, 2020.

[148] J.-B. Cordonnier, A. Loukas, and M. Jaggi, “On the relationship between self-attention and convolutional layers,” arXiv preprint arXiv:1911.03584, 2019.

[149] A. Liutkus, O. Cıfka, S.-L. Wu, U. Simsekli, Y.-H. Yang, and G. Richard, “Relative positional encoding for transformers with linear complexity,” in International Conference on Machine Learning (ICML’21), 2021.

[150] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, “Roformer: Enhanced transformer with rotary position embedding,” Neurocomputing, 2024.

[151] J. Li, X. Wang, Z. Tu, and M. R. Lyu, “On the diversity of multi-head attention,” Neurocomputing, 2021.

[152] J. Li, Z. Tu, B. Yang, M. R. Lyu, and T. Zhang, “Multi-head attention with disagreement regularization,” arXiv preprint arXiv:1810.10183, 2018.

[153] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “Gqa: Training generalized multi-query transformer models from multi-head checkpoints,” arXiv preprint arXiv:2305.13245, 2023.

[154] Z. Khan, M. Khaquan, O. Tafveez, B. Samiwala, and A. A. Raza, "Beyond uniform query distribution: Key-driven grouped query attention," arXiv preprint arXiv:2408.08454, 2024.

[155] Y. Chen, C. Zhang, X. Gao, R. D. Mullins, G. A. Constantinides, and Y. Zhao, "Optimised grouped-query attention mechanism for transformers," arXiv preprint arXiv:2406.14963, 2024.

[156] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," arXiv preprint arXiv:1606.08415, 2016.

[157] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," arXiv preprint arXiv:1701.06538, 2017.

[158] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," arXiv preprint arXiv:2006.16668, 2020.

[159] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand et al., "Mixtral of experts," arXiv preprint arXiv:2401.04088, 2024.

[160] S. Gupta, S. Mukherjee, K. Subudhi, E. Gonzalez, D. Jose, A. H. Awadallah, and J. Gao, "Sparsely activated mixture-of-experts are robust multi-task learners," arXiv preprint arXiv:2204.07689, 2022.

[161] B. Zoph, "Designing effective sparse expert models," in 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW'22), 2022.

[162] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale," in International conference on machine learning (ICML'22), 2022.

[163] OpenAI, "Openai platform: Models," https://platform.openai.com/docs/models, 2025.

[164] S. Pichai, D. Hassabis, and K. Kavukcuoglu, "Introducing gemini 2.0: Our new ai model for the agentic era," https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024, 2024.

[165] G. Team, R. Anil, S. Borgeaud, Y. Wu, J. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. Dai, A. Hauth et al., "Gemini: A family of highly capable multimodal models, 2024," arXiv preprint arXiv:2312.11805, 2024.

[166] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[167] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," Advances in neural information processing systems, 2019.

[168] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pretraining by representing and predicting spans," Transactions of the association for computational linguistics, 2020.

[169] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," arXiv preprint arXiv:1909.11942, 2019.

[170] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu et al., "Gpipe: Efficient training of giant neural networks using pipeline parallelism," Advances in neural information processing systems (NIPS'19), 2019.

[171] J. Sevilla, T. Besiroglu, O. Dudney, and A. Ho, "Report: The longest training run," https://epoch.ai/blog/the-longest-training-run, 2022.

[172] Bigscience, "Bigscience model training launched," https://bigscience.huggingface.co/blog/model-training-launched, 2022.

[173] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," arXiv preprint arXiv:2001.08361, 2020.

[174] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin et al., "Opt: Open pre-trained transformer language models," arXiv preprint arXiv:2205.01068, 2022.

[175] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," arXiv preprint arXiv:2109.01652, 2021.

[176] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," Advances in neural information processing systems, 2022.

[177] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., "Language models are unsupervised multitask learners," OpenAI blog, 2019.

[178] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer, and A. Gholami, "Kvquant: Towards 10 million context length llm inference with kv cache quantization," Advances in Neural Information Processing Systems, 2024.

[179] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval augmentation reduces hallucination in conversation," arXiv preprint arXiv:2104.07567, 2021.

[180] P. Béchard and O. M. Ayala, "Reducing hallucination in structured outputs via retrieval-augmented generation," arXiv preprint arXiv:2404.08189, 2024.

[181] B. Sarmah, D. Mehta, B. Hall, R. Rao, S. Patel, and S. Pasquali, "Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction," in Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF'24), 2024.

[182] Z. Jing, Y. Su, and Y. Han, "When large language models meet vector databases: A survey," in 2025 Conference on Artificial Intelligence x Multimedia (AIxMM), 2025.

[183] X. Zhao, X. Zhou, and G. Li, "Chat2data: An interactive data analysis system with rag, vector databases and llms," Proceedings of the VLDB Endowment, 2024.

[184] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Liu, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," arXiv preprint arXiv:2308.07107, 2023.

[185] Y. Liu, S. Yavuz, R. Meng, M. Moorthy, S. Joty, C. Xiong, and Y. Zhou, "Exploring the integration strategies of retriever and large language models," arXiv preprint arXiv:2308.12574, 2023.

[186] P. Xu, W. Ping, X. Wu, L. McAfee, C. Zhu, Z. Liu, S. Subramanian, E. Bakhturina, M. Shoeybi, and B. Catanzaro, "Retrieval meets long context large language models," arXiv preprint arXiv:2310.03025, 2023.

[187] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in Proceedings of the 10th ACM conference on recommender systems (RecSys'16), 2016.

[188] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir et al., "Wide & deep learning for recommender systems," in Proceedings of the 1st workshop on deep learning for recommender systems (DLRS'16), 2016.

[189] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston et al., "The youtube video recommendation system," in Proceedings of the fourth ACM conference on Recommender systems (RecSys'10), 2010.

[190] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in Proceedings of the 22nd international conference on Machine learning (ICML'05), 2005.

[191] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in Proceedings of the 24th international conference on Machine learning (ICML'07), 2007.

[192] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko et al., "Highly accurate protein structure prediction with alphafold," nature, 2021.

[193] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'16), 2016.

[194] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray et al., "Scaling laws for autoregressive generative modeling," arXiv preprint arXiv:2010.14701, 2020.

[195] K. Hong, G. Dai, J. Xu, Q. Mao, X. Li, J. Liu, K. Chen, Y. Dong, and Y. Wang, "Flashdecoding++: Faster large language model inference on gpus," arXiv preprint arXiv:2311.01282, 2023.

[196] L. Zhang, M. Wahib, H. Zhang, and S. Matsuoka, "A study of single and multi-device synchronization methods in nvidia gpus," in 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS'20), 2020.

[197] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang et al., "Large scale distributed deep networks," Advances in neural information processing systems, 2012.

[198] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.

[199] Z. Chen, L. Shi, X. Liu, J. Li, S. Liu, and Y. Xu, "Osp: Boosting distributed model training with 2-stage synchronization," in Proceedings of the 52nd International Conference on Parallel Processing (ICPP'23), 2023.

[200] Z. Tang, Z. Tang, J. Huang, X. Pan, R. Yan, Y. Wang, A. C. Zhou, S. Shi, X. Chu, and B. Li, "Dreamddp: Accelerating data parallel distributed llm training with layer-wise scheduled partial synchronization," arXiv preprint arXiv:2502.11058, 2025.

[201] A. Nabli, L. Fournier, P. Erbacher, L. Serrano, E. Belilovsky, and E. Oyallon, "Acco: Accumulate while you communicate, hiding communications in distributed llm training," 2024.

[202] X. Gu, K. Lyu, S. Arora, J. Zhang, and L. Huang, "A quadratic synchronization rule for distributed deep learning," arXiv preprint arXiv:2310.14423, 2023.

[203] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," Advances in neural information processing systems, 2017.

[204] S. Shi, X. Chu, and B. Li, "Mg-wfbp: Merging gradients wisely for efficient communication in distributed deep learning," IEEE Transactions on Parallel and Distributed Systems, 2021.

[205] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks." Proceedings of Machine Learning and Systems, 2019.

[206] B. Wang, Q. Xu, Z. Bian, and Y. You, "Tesseract: Parallelize the tensor parallelism efficiently," in Proceedings of the 51st International Conference on Parallel Processing (ICPP'22), 2022.

[207] K. Osawa, S. Li, and T. Hoefler, "Pipefisher: Efficient training of large language models using pipelining and fisher information matrices," Proceedings of Machine Learning and Systems, 2023.

[208] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in 2017 IEEE international symposium on high performance computer architecture (HPCA'17), 2017.

[209] C. He, S. Li, M. Soltanolkotabi, and S. Avestimehr, "Pipetransformer: Automated elastic pipelining for distributed training of transformers," arXiv preprint arXiv:2102.03161, 2021.

[210] Z. Hu, S. Shen, T. Bonato, S. Jeaugey, C. Alexander, E. Spada, J. Hammond, and T. Hoefler, "Demystifying nccl: An in-depth analysis of gpu communication protocols and algorithms," arXiv preprint arXiv:2507.04786, 2025.

[211] NVIDIA, "Nvidia collective communication library NCCL," 2025.

[212] MPI Forum, "Mpi: A message-passing interface standard," https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf, 1994.

[213] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," The International Journal of High Performance Computing Applications, 2005.

[214] J. Fei, C.-Y. Ho, A. N. Sahu, M. Canini, and A. Sapio, "Efficient sparse collective communication and its application to accelerate distributed deep learning," in Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM'21), 2021.

[215] G. Xu, Z. Le, Y. Chen, Z. Lin, Z. Jin, Y. Miao, and C. Li, "Autoccl: Automated collective communication tuning for accelerating distributed and parallel dnn training," in 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25), 2025.

[216] M. Zhai, J. He, Z. Ma, Z. Zong, R. Zhang, and J. Zhai, "Smartmoe: Efficiently training sparsely-activated models through combining offline and online parallelization," in 2023 USENIX Annual Technical Conference (USENIX ATC 23), 2023.

[217] H. Huang, N. Ardalani, A. Sun, L. Ke, H.-H. S. Lee, A. Sridhar, S. Bhosale, C.-J. Wu, and B. Lee, "Towards moe deployment: Mitigating inefficiencies in mixture-of-expert (moe) inference," arXiv preprint arXiv:2303.06182, 2023.

[218] S. Singh, O. Ruwase, A. A. Awan, S. Rajbhandari, Y. He, and A. Bhatele, "A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training," in Proceedings of the 37th International Conference on Supercomputing (ICS'23), 2023.

[219] W. Cai, J. Jiang, L. Qin, J. Cui, S. Kim, and J. Huang, "Shortcut-connected expert parallelism for accelerating mixture-of-experts," arXiv preprint arXiv:2404.05019, 2024.

[220] B. Wu, Y. Zhong, Z. Zhang, S. Liu, F. Liu, Y. Sun, G. Huang, X. Liu, and X. Jin, "Fast distributed inference serving for large language models," arXiv preprint arXiv:2305.05920, 2023.

[221] Z. Zheng, X. Ren, F. Xue, Y. Luo, X. Jiang, and Y. You, "Response length perception and sequence scheduling: An llm-empowered llm inference pipeline," Advances in Neural Information Processing Systems, 2023.

[222] R. Li, D. Fu, C. Shi, Z. Huang, and G. Lu, "Efficient llms training and inference: An introduction," IEEE Access, 2024.

[223] R. Y. Aminabadi, S. Rajbhandari, A. A. Awan, C. Li, D. Li, E. Zheng, O. Ruwase, S. Smith, M. Zhang, J. Rasley et al., "Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale," in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC'22), 2022.

[224] P. Chaturvedi, A. Khan, M. Tian, E. Huerta, and H. Zheng, "Inference-optimized ai and high performance computing for gravitational wave detection at scale," Frontiers in Artificial Intelligence, 2022.

[225] S. Verma and N. Vaidya, “Mastering llm techniques: Inference optimization,” Retrieved May, 2023.

[226] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, “Splitwise: Efficient generative llm inference using phase splitting,” in 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA’24), 2024.

[227] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, “Dense passage retrieval for open-domain question answering.” in Conference on Empirical Methods in Natural Language Processing (EMNLP’20), 2020.

[228] A. Mansurova, A. Mansurova, and A. Nugumanova, “Qa-rag: Exploring llm reliance on external knowledge,” Big Data and Cognitive Computing, 2024.

[229] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, “Retrieval-augmented generation for large language models: A survey,” arXiv preprint arXiv:2312.10997, 2023.

[230] D. Quinn, M. Nouri, N. Patel, J. Salihu, A. Salemi, S. Lee, H. Zamani, and M. Alian, “Accelerating retrieval-augmented generation,” in Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’25), 2025.

[231] NVIDIA, “Nvidia opens nvlink for custom silicon integration.” 2025.

[232] ——, “Nvidia nvlink-c2c: Extending nvlink to chip-level integration.” 2025.

[233] IEEE, “Ieee standard for ethernet,” 2025.

[234] J. F. Shoch and J. A. Hupp, “Measured performance of an ethernet local network,” Communications of the ACM, 1980.

[235] B. Lowekamp, D. O’Hallaron, and T. Gross, “Topology discovery for large ethernet networks,” ACM SIGCOMM Computer Communication Review, 2001.

[236] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, “Past: Scalable ethernet for data centers,” in Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT’12), 2012.

[237] InfiniBand Trade Association, “Infiniband architecture specification volume 1 release 1.2.1,” 2007.

[238] G. F. Pfister, “An introduction to the infiniband architecture,” High performance mass storage and parallel I/O, 2001.

[239] T. Hoefler, T. Schneider, and A. Lumsdaine, “Optimized routing for large-scale infiniband networks,” in 2009 17th IEEE Symposium on High Performance Interconnects (HOTI’09), 2009.

[240] K. Hintze, S. Graham, S. Dunlap, and P. Sweeney, “Infiniband network monitoring: Challenges and possibilities,” in International Conference on Critical Infrastructure Protection (ICCIP’21), 2021.

[241] V. A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, “Reducing activation recomputation in large transformer models,” Proceedings of Machine Learning and Systems, 2023.

[242] H. Liu, M. Zaharia, and P. Abbeel, “Ring attention with blockwise transformers for near-infinite context,” arXiv preprint arXiv:2310.01889, 2023.

[243] A. Yang, J. Yang, A. Ibrahim, X. Xie, B. Tang, G. Sizov, J. Reizenstein, J. Park, and J. Huang, “Context parallelism for scalable million-token inference,” arXiv preprint arXiv:2411.01783, 2024.

[244] I. Goldwasser, H. Petty, P. Desale, and K. Devleker, “Nvidia gb200 nvl72 delivers trillion-parameter llm training and real-time inference,” https://developer.nvidia.com/blog/nvidia-gb200-nvl72-delivers-trillion-parameter-llm-training-and-real-time-inference/, 2024.

[245] Z. Wang, L. Luo, Q. Ning, C. Zeng, W. Li, X. Wan, P. Xie, T. Feng, K. Cheng, X. Geng et al., “Srnic: A scalable architecture for rdmanics,” in 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI’23), 2023.

[246] F. Daoud, A. Watad, and M. Silberstein, “Gpurdma: Gpu-side library for high performance networking from gpu kernels,” in Proceedings of the 6th international Workshop on Runtime and Operating Systems for Supercomputers (ROSS’16), 2016.

[247] E. Agostini, D. Rossetti, and S. Potluri, “Gpudirect async: Exploring gpu synchronous communication techniques for infiniband clusters,” Journal of Parallel and Distributed Computing, 2018.

[248] P. Thinakaran, J. Raj, B. Sharma, M. T. Kandemir, and C. R. Das, “The curious case of container orchestration and scheduling in gpu-based datacenters,” in Proceedings of the ACM symposium on cloud computing, 2018.

[249] S. Lockhart, A. Bienz, W. D. Gropp, and L. N. Olson, "Characterizing the performance of node-aware strategies for irregular point-to-point communication on heterogeneous architectures," Parallel Computing, 2023.

[250] J. Networks, "Ai data center network with juniper apstra, amd gpus, and vast storage—juniper validated design (jvd)," https://www.juniper.net/documentation/us/en/software/jvd/jvd-ai-dc-apstra-amd/jvd-ai-dc-apstra-amd-vast.pdf, 2025.

[251] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," ACM SIGARCH computer architecture news, 2009.

[252] K. Lim, Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, and T. F. Wenisch, "System-level implications of disaggregated memory," in IEEE International Symposium on High-Performance Comp Architecture (HPCA'12), 2012.

[253] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in 12th USENIX symposium on operating systems design and implementation (OSDI'16), 2016.

[254] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, "Network support for resource disaggregation in next-generation datacenters," in Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks (HotNets'13), 2013.

[255] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VI2: A scalable and flexible data center network," in Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM'09), 2009.

[256] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," ACM SIGCOMM computer communication review, 2008.

[257] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12), 2012.

[258] C. Tang, "Meta' s hyperscale infrastructure: Overview and insights," Communications of the ACM, 2025.

[259] A. Andreyev, W. Xu, and A. Eckert, "Reinventing facebook's data center network," https://engineering.fb.com/2019/03/14/data-center-engineering/f16-minipack/, 2019.

[260] Open Compute Project, “Open rack v3 bbu module specification v1.4,” https://www.opencompute.org/documents/open-rack-v3-bbu-module-spec-1-4-pdf, 2023.

[261] International Electrotechnical Commission, “Iec 60297-3-110:2018: Mechanical structures for electrical and electronic equipment - dimensions of mechanical structures of the 482,6 mm (19 in) series - part 3-110: Residential racks and cabinets for smart houses,” https://webstore.iec.ch/en/publication/32626, 2018.

[262] NVIDIA, “Nvidia gb200 nvl72: Powering the new era of computing,” https://www.nvidia.com/en-us/data-center/gb200-nvl72, 2025.

[263] ——, “Nvidia gb300 nvl72: Built for the age of ai reasoning,” https://www.nvidia.com/en-us/data-center/gb300-nvl72/, 2025.

[264] Micron, “Hbm3e: The industry’s fastest, highest-capacity high-bandwidth memory (hbm) to advance generative ai innovation,” https://www.micron.com/products/memory/hbm/hbm3e, 2023.

[265] NVIDIA, “Nvidia grace cpu superchip whitepaper: Performance and efficiency for the mordern data center,” https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-cpu-superchip, 2024.

[266] ——, “Nvidia gh200 grace hopper superchip architecture whitepaper: Performance and productivity for strong-scaling hpc and giant ai workloads,” https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper, 2024.

[267] ——, “Nvidia grace cpu superchip datasheet: The world’s first no-compromise data center cpu,” https://resources.nvidia.com/en-us-grace-cpu/data-center-datasheet?ncid=so-yout-933463-vt37, 2024.

[268] Y. Wei, Y. C. Huang, H. Tang, N. Sankaran, I. Chadha, D. Dai, O. Oluwole, V. Balan, and E. Lee, “9.3 nvlink-c2c: A coherent off package chip-to-chip interconnect with 40gbps/pin single-ended signaling,” in 2023 IEEE International Solid-State Circuits Conference (ISSCC’23), 2023.

[269] G. Schieffer, J. Wahlgren, J. Ren, J. Faj, and I. Peng, “Harnessing integrated cpu-gpu system memory for hpc: a first look into grace hopper,” in Proceedings of the 53rd International Conference on Parallel Processing (ICPP’24), 2024.

[270] F. Werner, M. Weisgut, and T. Rabl, “Towards memory disaggregation via nvlink c2c: Benchmarking cpu-requested gpu memory access,” in Proceedings of the 4th Workshop on Heterogeneous Composable and Disaggregated Systems (HCDS’25), 2025.

[271] I. Burstein, “Nvidia data center processing unit (dpu) architecture,” in 2021 IEEE Hot Chips 33 Symposium (HCS’21), 2021.

[272] NVIDIA, “Nvidia bludefield-3 networking platform datasheet: The 400gb/s infrastructure compute platform for powering the world’s data cneters,” https://resources.nvidia.com/en-us-accelerated-networking-resource-library/datasheet-nvidia-bluefield, 2023.

[273] ——, “Nvidia connectx-7 400g adpaters: Accelerated networking for mordern data center infrastructures,” https://resources.nvidia.com/en-us-accelerated-networking-resource-library/connectx-7-datasheet, 2021.

[274] ——, “Nvidia connectx-8 supernic datasheet,” https://resources.nvidia.com/en-us-accelerated-networking-resource-library/connectx-datasheet-c, 2025.

[275] B. Hedlund, “Top of rack vs end of row data center designs,” https://bradhedlund.com/2009/04/05/top-of-rack-vs-end-of-row-data-center-designs/, 2009.

[276] FS, “Top of rack and end of row: What’s the difference?” https://community.fs.com/article/tor-vs-eor-data-center-architecture-design.html, 2021.

[277] S. Liu, Q. Cheng, A. Wonfor, R. Penty, I. White, and P. M. Watts, “A low latency optical top of rack switch for data centre networks with minimized processor energy load,” in Optical Fiber Communication Conference (OFC’14), 2014.

[278] A. S. Alhazmi, O. Z. Aletri, T. E. El-Gorashi, M. T. Alresheedi, and J. M. Elmirghani, “Data center top of rack switch to multiple spine switches optical wireless uplinks,” in 2020 22nd International Conference on Transparent Optical Networks (ICTON), 2020.

[279] A. Singla, P. B. Godfrey, and A. Kolla, “High throughput data center topology design,” in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI’14), 2014.

[280] FS Inc., “What is an aggregate switch and how to choose?” https://www.fs.com/blog/what-is-an-aggregate-switch-1340.html, 2023.

[281] Y. Li, D. Wei, X. Chen, Z. Song, R. Wu, Y. Li, X. Jin, and W. Xu, “Dumbnet: A smart data center network fabric with dumb switches,” in Proceedings of the Thirteenth EuroSys Conference (EuroSys’18), 2018.

[282] Cisco, “Cisco aci multi-tier architecture whitepaper,” https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-742214.pdf, 2024.

[283] HPE, “What is spine-leaf architecture?” https://www.hpe.com/emea_africa/en/what-is/spine-leaf-architecture.html, 2024.

[284] M. Alizadeh and T. Edsall, “On the data path performance of leaf-spine datacenter fabrics,” in 2013 IEEE 21st annual symposium on high-performance interconnects (HOTI’13), 2013.

[285] NVIDIA Corporation, “Dgx superpod reference architecture with b200,” https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-b200/latest/index.html, 2024.

[286] ——, “Dgx superpod reference architecture with gb200,” https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-gb200/latest/index.html, 2024.

[287] NVIDIA, “Nvidia quantum-2 infiniband platform datasheet: Extreme performance for exascale ai,” https://nvdam.widen.net/s/dps8txlsrf/infiniband-ndr-400g-architecture-datasheet-1620877-r4, 2022.

[288] ——, “Nvidia quantum-x800 infiniband switches: Accelerate ai workloads with 8000g infiniband,” https://nvdam.widen.net/s/nfdzskhmnc/infiniband-datasheet-quantum-family-3231555, 2025.

[289] ——, “Nvidia spectrum-x datasheet: Purpose-built for ethernet ai clouds,” https://resources.nvidia.com/en-us-networking-ai/networking-ethernet-1, 2023.

[290] N. Rasmussen and W. Torell, “Data center projects: establishing a floor plan,” Whitepaper, 2007.

[291] Vertiv Corporation, “Understanding coolant distribution units (cdus) for liquid cooling,” https://www.vertiv.com/en-us/about/news-and-insights/articles/educational-articles/understanding-coolant-distribution-units-cdus-for-liquid-cooling/, n.d.

[292] BOYD, “Data center cooling systems: Coolant distribution unit liquid cooling,” https://www.boydcorp.com/blog/data-center-cooling-systems-coolant-distribution-unit-liquid-cooling.html, 2024.

[293] S. Kala and S. Mills, “Dc power distribution unit for v2 open rack,” https://www.opencompute.org/wiki/Open_Rack/SpecsAndDesigns, 2015.

[294] H. Keyhani, “Open rack v3 48v psu specification rev 1.0,” https://www.opencompute.org/wiki/Open_Rack/SpecsAndDesigns, 2022.

[295] N. Blach, M. Besta, D. De Sensi, J. Domke, H. Harake, S. Li, P. Iff, M. Konieczny, K. Lakhotia, A. Kubicek et al., "A high-performance design, implementation, deployment, and evaluation of the slim fly network," in 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24), 2024.

[296] J. Liu, W. Jiang, P. Wyckoff, D. K. Panda, D. Ashton, D. Buntinas, W. Gropp, and B. Toonen, "Design and implementation of mpich2 over infiniband with rdma support," in 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004.

[297] D. De Sensi, L. Pichetti, F. Vella, T. De Matteis, Z. Ren, L. Fusco, M. Turisini, D. Cesarini, K. Lust, A. Trivedi et al., "Exploring gpu-to-gpu communication: Insights into supercomputer interconnects," in SC24: International Conference for High Performance Computing, Networking, Storage and Analysis (SC'24), 2024.

[298] B. Lebiednik, A. Mangal, and N. Tiwari, "A survey and evaluation of data center network topologies," arXiv preprint arXiv:1605.01701, 2016.

[299] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM'09), 2009.

[300] H. Xu, C. Feng, and B. Li, "Temperature aware workload management in geo-distributed datacenters," 2013.

[301] Baxtel, "Global data center map," https://baxtel.com/map, 2025.

[302] AWS, "Global infrastructure regions and availability zones," https://aws.amazon.com/about-aws/global-infrastructure/regions_az/, 2024.

[303] Microsoft, "Azure global infrastructure: Grow with confidence and choice on a trusted, sustainable, and advanced cloud and ai infrastructure." https://azure.microsoft.com/en-us/explore/global-infrastructure, 2025.

[304] ——, "What are azure availability zones?" https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview?tabs=azure-cli, 2025.

[305] Meta, "Global data center fleet: Connection starts with community," https://datacenters.atmeta.com/all-locations/, 2025.

[306] Google, "Global locations - regions and zones," https://cloud.google.com/about/locations, 2025.

[307] J. Romero, J. Yin, N. Laanait, B. Xie, M. T. Young, S. Treichler, V. Starchenko, A. Borisevich, A. Sergeev, and M. Matheson, "Accelerating collective communication in data parallel training across deep learning frameworks," in 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22), 2022.

[308] S. Shi, X. Chu, and B. Li, "Exploiting simultaneous communications to accelerate data parallel distributed deep learning," in IEEE INFOCOM 2021-IEEE Conference on Computer Communications (INFOCOM'21), 2021.

[309] J. Fang, H. Fu, G. Yang, and C.-J. Hsieh, "Redsync: reducing synchronization bandwidth for distributed deep learning training system," Journal of Parallel and Distributed Computing, 2019.

[310] N. Xie, T. Norman, D. Grewe, and D. Vytiniotis, "Synthesizing optimal parallelism placement and reduction strategies on hierarchical systems for deep learning," Proceedings of Machine Learning and Systems, 2022.

[311] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns." in 15th edition of the Interspeech Conference (Interspeech'14), 2014.

[312] N. Alnaasan, A. Jain, A. Shafi, H. Subramoni, and D. K. Panda, "Accdp: accelerated data-parallel distributed dnn training for modern gpu-based hpc clusters," in 2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC'22), 2022.

[313] M. Cho, U. Finkler, D. Kung, and H. Hunter, "Blueconnect: Decomposing all-reduce for deep learning on heterogeneous network hierarchy," Proceedings of Machine Learning and Systems, 2019.

[314] S. Legtchenko, H. Williams, K. Razavi, A. Donnelly, R. Black, A. Douglas, N. Cheriere, D. Fryer, K. Mast, A. D. Brown et al., "Understanding rack-scale disaggregated storage," in 9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'17), 2017.

[315] J. Kundu, W. Guo, A. BanaGozar, U. De Alwis, S. Sengupta, P. Gupta, and A. Mallik, "Performance modeling and workload analysis of distributed large language model training and inference," in 2024 IEEE International Symposium on Workload Characterization (IISWC'24), 2024.

[316] Q. Hu, Z. Ye, Z. Wang, G. Wang, M. Zhang, Q. Chen, P. Sun, D. Lin, X. Wang, Y. Luo et al., "Characterization of large language model development in the datacenter," in 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24), 2024.

[317] M. Sponner, B. Waschneck, and A. Kumar, “Ai-driven performance modeling for ai inference workloads,” Electronics, 2022.

[318] W. Jiang, S. Subramanian, C. Graves, G. Alonso, A. Yazdanbakhsh, and V. Dadu, “Rago: Systematic performance optimization for retrieval-augmented generation serving,” arXiv preprint arXiv:2503.14649, 2025.

[319] B. Sharma, L. Yang, and H. Pham, “Multi-tenancy for ai inference at meta scale: Designed for engineers that manage large-scale information systems serving millions of people. the operation of large-scale systems often introduces complex, unprecedented engineering challenges,” https://atscaleconference.com/multi-tenancy-for-ai-inference-at-meta-scale/, 2023.

[320] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., “Llama 2: Open foundation and fine-tuned chat models,” arXiv preprint arXiv:2307.09288, 2023.

[321] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, “Llm inference serving: Survey of recent advances and opportunities,” in 2024 IEEE High Performance Extreme Computing Conference (HPEC), 2024.

[322] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li et al., “A survey on efficient inference for large language models,” arXiv preprint arXiv:2404.14294, 2024.

[323] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, “From local to global: A graph rag approach to query-focused summarization,” arXiv preprint arXiv:2404.16130, 2024.

[324] G. Liu, H. Yin, B. Zhu, J. Chen, C.-W. Ngo, and Y.-G. Jiang, “Retrieval augmented recipe generation,” in 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV’25), 2025.

[325] F. Liang, Z. Zhang, H. Lu, V. Leung, Y. Guo, and X. Hu, “Communication-efficient large-scale distributed deep learning: A comprehensive survey,” arXiv preprint arXiv:2404.06114, 2024.

[326] W. Xiao, S. Ren, Y. Li, Y. Zhang, P. Hou, Z. Li, Y. Feng, W. Lin, and Y. Jia, “Antman: Dynamic scaling on gpu clusters for deep learning,” in 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI’20), 2020.

[327] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, “Ai and memory wall,” IEEE Micro, 2024.

[328] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, “Efficient memory disaggregation with infiniswap,” in 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI’17), 2017.

[329] P. Zuo, J. Sun, L. Yang, S. Zhang, and Y. Hua, “One-sided rdma-conscious extendible hashing for disaggregated memory,” in 2021 USENIX Annual Technical Conference (USENIX ATC 21), 2021.

[330] I. Calciu, M. T. Imran, I. Puddu, S. Kashyap, H. A. Maruf, O. Mutlu, and A. Kolli, “Rethinking software runtimes for disaggregated memory,” in Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’21), 2021.

[331] Z. Wang, X. Wei, J. Gu, H. Xie, R. Chen, and H. Chen, “Odrp: On-demand remote paging with programmable rdma,” in 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI’25), 2025.

[332] J. Fan, X. Ye, J. Kim, B. Archambeault, and A. Orlandi, “Signal integrity design for high-speed digital circuits: Progress and directions,” IEEE Transactions on Electromagnetic Compatibility, 2010.

[333] S. H. Hall and H. L. Heck, Advanced signal integrity for high-speed digital designs, 2011.

[334] J. Wahlgren, M. Gokhale, and I. B. Peng, “Evaluating emerging cxl-enabled memory pooling for hpc systems,” in 2022 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC), 2022.

[335] R. Stenfort, J. Adrian, and M. Sompura, “Overview of edsff e1.s form factors,” https://americas.kioxia.com/content/dam/kioxia/en-us/business/ssd/data-center-ssd/asset/KIOXIA_Meta_Microsoft_EDSFF_E1_S_Intro_White_Paper.pdf, 2023.

[336] SNIA, “Enterprise and data center ssd form factor - the end of the 2.5in disk era?” https://www.snia.org/sites/default/files/SSSI/EDSFF%20Webcast%208-4-2020%20fnl.pdf, 2020.

[337] J. Hands, “Edsff - a dynamic family of form factors for data center ssds,” https://www.snia.org/sites/default/files/SSSI/OCP%20EDSFF%20JM%20Hands.pdf, 2020.

[338] KIOXIA, “Edsff - a new ssd form factor for next gen servers and storage,” https://americas.kioxia.com/en-us/business/ssd/solution/edsff.html, 2025.

[339] B. Jacob, S. W. Ng, and D. T. Wang, “Chapter 10 - dram memory system organization,” in Memory Systems, 2008.

[340] J. Schneider and I. Smalley, “What is a dual in-line memory module (dimm)?” https://www.ibm.com/think/topics/dimm, 2024.

[341] J. Kanade, “What is a dual in-line memory module (dimm)? meaning, characteristics, and types,” https://www.spiceworks.com/tech/tech-general/articles/what-is-dimm/, 2023.

[342] D. M. Harris and S. L. Harris, “8 - memory and i/o systems,” in Digital Design and Computer Architecture (Second Edition), 2013.

[343] X. Wang, J. Wang, X. Tao, M. Yang, and J. Lai, “Design and implementation of ddr3 sdram controller,” in 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT’18), 2018.

[344] U. Kang, H.-J. Chung, S. Heo, D.-H. Park, H. Lee, J. H. Kim, S.-H. Ahn, S.-H. Cha, J. Ahn, D. Kwon et al., “8 gb 3-d ddr3 dram using through-silicon-via technology,” IEEE Journal of Solid-State Circuits, 2009.

[345] Samsung Electronics, “Ddr3 sdram specification,” https://download.semiconductor.samsung.com/resources/data-sheet/877094ds_ddr3_1gb_d-die_based_udimm_rev123.pdf, 2009.

[346] M. A. Islam, M. Y. Arafath, and M. J. Hasan, “Design of ddr4 sdram controller,” in 8th International Conference on Electrical and Computer Engineering (ICEMCE’14), 2014.

[347] S. Lee, H. Cho, Y. H. Son, Y. Ro, N. S. Kim, and J. H. Ahn, “Leveraging power-performance relationship of energy-efficient modern dram devices,” IEEE Access, 2018.

[348] Micron, “Tn-40-40: Ddr4 point-to-point design guide,” https://www.mouser.com/pdfDocs/Micron_DDR4_Design_Guide.pdf?srsltid=AfmBOorGptc-D9pZ_YIZMJiEJxCf6Aq83LKLWoVmlwFafNuPmxixFins, 2020.

[349] M. H. Hajkazemi, M. K. Tavana, and H. Homayoun, “Wide i/o or lpddr? exploration and analysis of performance, power and temperature trade-offs of emerging dram technologies in embedded mpsocs,” in 2015 33rd IEEE International Conference on Computer Design (ICCD’15), 2015.

[350] K.-S. Ha, C.-K. Lee, D. Lee, D. Moon, H.-R. Hwang, D. Park, Y.-H. Kim, Y. H. Son, B. Na, S. Lee et al., “A 7.5 gb/s/pin 8-gb lpddr5 sdram with various high-speed and low-power techniques,” IEEE Journal of Solid-State Circuits, 2019.

[351] Y. Seo, J. Choi, S. Cho, H. Han, W. Kim, G. Ryu, J. Ahn, Y. Cho, S. Choi, S. Lee et al., “13.8 a 1a-nm 1.05 v 10.5 gb/s/pin 16gb lpddr5 turbo dram with wck correction strategy, a voltage-offset-calibrated receiver and parasitic capacitance reduction,” in 2024 IEEE International Solid-State Circuits Conference (ISSCC), 2024.

[352] C. Clos, “A study of non-blocking switching networks,” Bell System Technical Journal, 1953.

[353] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano et al., “Jupiter rising: A decade of clos topologies and centralized control in google’s datacenter network,” ACM SIGCOMM computer communication review, 2015.

[354] H. Choo, S.-M. Yoo, and H. Y. Youn, “Processor scheduling and allocation for 3d torus multicomputer systems,” IEEE Transactions on Parallel and Distributed Systems, 2000.

[355] P. López and J. Duato, “Deadlock-free adaptive routing algorithms for the 3d-torus: Limitations and solutions,” in International Conference on Parallel Architectures and Languages Europe (PARLE’93), 1993.

[356] P. Costa, A. Donnelly, G. O’Shea, and A. Rowstron, “Camcubeos: a key-based network stack for 3d torus cluster topologies,” in Proceedings of the 22nd international symposium on High-performance parallel and distributed computing (HPDC’13), 2013.

[357] S. Cheng, W. Zhong, K. E. Isaacs, and K. Mueller, “Visualizing the topology and data traffic of multi-dimensional torus interconnect networks,” IEEE Access, 2018.

[358] J. Kim, W. J. Dally, S. Scott, and D. Abts, “Technology-driven, highly-scalable dragonfly topology,” ACM SIGARCH Computer Architecture News, 2008.

[359] J. Kim, W. Dally, S. Scott, and D. Abts, “Cost-efficient dragonfly topology for large-scale systems,” IEEE micro, 2009.

[360] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, “Dragonfly+: Low cost topology for scaling datacenters,” in 2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB’17), 2017.

[361] Vortex, “Vortex: Opencl compatible risc-v gpgpu,” https://vortex.cc.gatech.edu/, 2025.

[362] B. Tine, K. P. Yalamarthy, F. Elsabbagh, and K. Hyesoon, “Vortex: Extending the risc-v isa for gpgpu and 3d-graphics,” in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (Micro’21), 2021.

[363] E. Cui, T. Li, and Q. Wei, “Risc-v instruction set architecture extensions: A survey,” IEEE Access, 2023.

[364] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, “The risc-v instruction set manual, volume i: User-level isa, version 2.0,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, 2014.

[365] C. Lameter, “Numa (non-uniform memory access): An overview: Numa becomes more common because memory controllers get close to execution units on microprocessors.” Queue, 2013.

[366] N. Denoyelle, B. Goglin, A. Ilic, E. Jeannot, and L. Sousa, “Modeling non-uniform memory access on large compute nodes with the cache-aware roofline model,” IEEE Transactions on Parallel and Distributed Systems, 2018.

[367] Z. Majo and T. R. Gross, “(mis) understanding the numa memory system performance of multithreaded workloads,” in 2013 IEEE International Symposium on Workload Characterization (IISWC’13), 2013.

[368] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark et al., “Learning transferable visual models from natural language supervision,” in International conference on machine learning (ICML’21), 2021.

[369] H. Shi, M. Hayat, Y. Wu, and J. Cai, “Proposalclip: Unsupervised open-category object proposal generation via exploiting clip cues,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR’22), 2022.

[370] Y. Ma, G. Xu, X. Sun, M. Yan, J. Zhang, and R. Ji, “X-clip: End-to-end multi-grained contrastive learning for video-text retrieval,” in Proceedings of the 30th ACM international conference on multimedia (MM’22), 2022.

[371] P. Ristoski and H. Paulheim, “Rdf2vec: Rdf graph embeddings for data mining,” in International semantic web conference (ISWC’16), 2016.

[372] M. Cochez, P. Ristoski, S. P. Ponzetto, and H. Paulheim, “Global rdf vector space embeddings,” in International semantic web conference (ISWC’17), 2017.

[373] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” IEEE transactions on neural networks and learning systems, 2020.

[374] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE transactions on neural networks, 2008.

[375] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” AI open, 2020.

[376] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” Information Systems, 2014.

[377] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” IEEE transactions on pattern analysis and machine intelligence, 2018.

[378] X. Xu, C. Li, Y. Wang, and Y. Xia, “Multiattribute approximate nearest neighbor search based on navigable small world graph,” Concurrency and Computation: Practice and Experience, 2020.

[379] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, “The faiss library,” arXiv preprint arXiv:2401.08281, 2024.

[380] D. Danopoulos, C. Kachris, and D. Soudris, “Approximate similarity search with faiss framework using fpgas on the cloud,” in International Conference on Embedded Computer Systems (SAMOS’19), 2019.

[381] J. Mohoney, A. Pacaci, S. R. Chowdhury, A. Mousavi, I. F. Ilyas, U. F. Minhas, J. Pound, and T. Rekatsinas, “High-throughput vector similarity search in knowledge graphs,” Proceedings of the ACM on Management of Data, 2023.

[382] C. Mavromatis and G. Karypis, “Gnn-rag: Graph neural retrieval for large language model reasoning,” arXiv preprint arXiv:2405.20139, 2024.

[383] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, “Grag: Graph retrieval-augmented generation,” arXiv preprint arXiv:2405.16506, 2024.

[384] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini et al., “Deep learning recommendation model for personalization and recommendation systems,” arXiv preprint arXiv:1906.00091, 2019.

[385] S. Liu, N. Zheng, H. Kang, X. Simmons, J. Zhang, M. Langer, W. Zhu, M. Lee, and Z. Wang, “Embedding optimization for training large-scale deep learning recommendation systems with embark,” in Proceedings of the 18th ACM Conference on Recommender Systems (RecSys’24), 2024.

[386] D. Mudigere, Y. Hao, J. Huang, Z. Jia, A. Tulloch, S. Sridharan, X. Liu, M. Ozdal, J. Nie, J. Park et al., “Software-hardware co-design for fast and scalable training of deep learning recommendation models,” in Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA’22), 2022.

[387] G. Bosilca, T. Herault, A. Rezmerita, and J. Dongarra, “On scalability for mpi runtime systems,” in 2011 IEEE International Conference on Cluster Computing (CLUSTER’11), 2011.

[388] G. M. Shipman, T. S. Woodall, R. L. Graham, A. B. Maccabe, and P. G. Bridges, "Infiniband scalability in open mpi," in Proceedings 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS'06), 2006.

[389] J. Huang, K. Ouyang, Y. Zhai, J. Liu, M. Si, K. Raffenetti, H. Zhou, A. Hori, Z. Chen, Y. Guo et al., "Accelerating mpi collectives with process-in-process-based multi-object techniques," in Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC'23), 2023.

[390] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng et al., "Warp-x: A new exascale computing platform for beam–plasma simulations," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2018.

[391] P. C. Liewer and V. K. Decyk, "A general concurrent algorithm for plasma particle-in-cell simulation codes," Journal of Computational Physics, 1989.

[392] UCLA Plasma Simulation Group, "Particle-in-cell skeleton codes," https://github.com/UCLA-Plasma-Simulation-Group/PIC-skeleton-codes, 2017.

[393] P. R. Spalart and V. Venkatakrishnan, "On the role and challenges of cfd in the aerospace industry," The Aeronautical Journal, 2016.

[394] M. Karp, E. Suarez, J. H. Meinke, M. I. Andersson, P. Schlatter, S. Markidis, and N. Jansson, "Experience and analysis of scalable high-fidelity computational fluid dynamics on modular supercomputing architectures," The international journal of high performance computing applications, 2025.

[395] OpenFOAM, "Openfoam v13," https://openfoam.org/, 2024.

[396] C. Petersen, "Building the case for ualink: A dedicated scale-up memory semantic fabric," https://www.asteralabs.com/building-the-case-for-ualink-a-dedicated-scale-up-memory-semantic-fabric/, 2024.

[397] J. Ames and R. Lowman, "How ultra ethernet and ualink enable high-performance, scalable ai nntworks," https://www.synopsys.com/articles/ultra-ethernet-ualink-ai-networks.html#5, 2025.

[398] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn, "Collective communication: theory, practice, and experience," Concurrency and Computation: Practice and Experience, 2007.

[399] J. Bruck, C.-T. Ho, S. Kipnis, and D. Weathersby, "Efficient algorithms for all-to-all communications in multi-port message-passing systems," in Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures (SPAA'94), 1994.

[400] C. Lutz, S. Breß, S. Zeuch, T. Rabl, and V. Markl, "Pump up the volume: Processing large data on gpus with fast interconnects," in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGCOMM'20), 2020.

[401] Y. Zhang, R. Nazaraliyev, S. B. Dutta, A. Marquez, K. Barker, and N. Abu-Ghazaleh, "Nvbleed: Covert and side-channel attacks on nvidia multi-gpu interconnect," arXiv preprint arXiv:2503.17847, 2025.

[402] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," ACM SIGARCH Computer Architecture News, 2015.

[403] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, H. Abdel-Aziz et al., "Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc," in 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA'21), 2021.

[404] A. Dhar, C. Thorens, L. M. Lazier, and L. Cavigelli, "Ascend-cc: Confidential computing on heterogeneous npu for emerging generative ai workloads," arXiv preprint arXiv:2407.11888, 2024.

[405] J. Coburn, C. Tang, S. A. Asal, N. Agrawal, R. Chinta, H. Dixit, B. Dodds, S. Dwarakapuram, A. Firoozshahian, C. Gao et al., "Meta's second generation ai chip: Model-chip co-design and productionization experiences," in Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA'24), 2025.

[406] A. Firoozshahian, J. Coburn, R. Levenstein, R. Nattoji, A. Kamath, O. Wu, G. Grewal, H. Aepala, B. Jakka, B. Dreyer, A. Hutchin, U. Diril, K. Nair, E. K. Aredestani, M. Schatz, Y. Hao, R. Komuravelli, K. Ho, S. Abu Asal, J. Shajrawi, K. Quinn, N. Sreedhara, P. Kansal, W. Wei, D. Jayaraman, L. Cheng, P. Chopda, E. Wang, A. Bikumandla, A. Karthik Sengottuvel, K. Thottempudi, A. Narasimha, B. Dodds, C. Gao, J. Zhang, M. Al-Sanabani, A. Zehtabioskuie, J. Fix, H. Yu, R. Li, K. Gondkar, J. Montgomery, M. Tsai, S. Dwarakapuram, S. Desai, N. Avidan, P. Ramani, K. Narayanan, A. Mathews, S. Gopal, M. Naumov, V. Rao, K. Noru, H. Reddy, P. Venkatapuram, and A. Bjorlin, "Mtia: First generation silicon targeting meta's recommendation systems," in Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA'23), 2023.

[407] N. Bshara, “Aws trainium: the journey for designing and optimization full stack ml hardware,” in Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’24), 2024.

[408] AWS, “Trainium,” https://aws.amazon.com/ko/ai/machine-learning/trainium/, 2024.

[409] G. Hutt, V. Viswanathan, and A. Nadolski, “Deliver high performance ml inference with aws inferentia,” https://d1.awsstatic.com/events/reinvent/2019/REPEAT_1_Deliver_high_performance_ML_inference_with_AWS_Inferentia_CMP324-R1.pdf, 2019.

[410] aws, “Inferentia,” https://aws.amazon.com/ko/ai/machine-learning/inferentia/, 2024.

[411] S. Xu and C. Ramakrishnan, “Inside maia 100,” in Hot Chips 36 Symposium (HCS’24), 2024.

[412] Microsoft, “Azure maia for the era of ai: From silicon to software to systems,” https://azure.microsoft.com/en-us/blog/azure-maia-for-the-era-of-ai-from-silicon-to-software-to-systems/, 2024.

[413] R. Kaplan, “Intel gaudi 3 ai accelerator: Architected for gen ai training and inference,” in 2024 IEEE Hot Chips 36 Symposium (HCS), 2024.

[414] Intel, “Intel gaudi 3 ai accelerator whitepaper,” https://www.intel.com/content/www/us/en/content-details/817486/intel-gaudi-3-ai-accelerator-white-paper.html, 2025.

[415] UCIe Consortium, “Ucie 2.0 specification,” https://www.uciexpress.org/2-0-spec-download, 2024.

panmnesia

Notice and Disclaimer

- Panmnesia, the Panmnesia logo, and other Panmnesia marks are trademarks of Panmnesia, Inc. or its subsidiaries. Other names and brands may be claimed as the property of others.
- All content contained in this document is protected by applicable copyright laws. Any unauthorized use, reproduction, distribution, or transmission of the content is strictly prohibited without prior written consent of Panmnesia.
- All information included herein is provided "AS IS." Panmnesia hereby disclaims all warranties, representations, and guarantees of any kind with respect to the information in this document, including without limitation, warranties of merchantability, non-infringement, accuracy, completeness, timeliness, or fitness for any particular purpose.
- Panmnesia reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.
- Neither Panmnesia nor any of its affiliates, officers, employees, or representatives shall bear any responsibility or liability whatsoever for any errors, omissions, or consequences arising from the use of or reliance upon any information included herein. Any recipient should conduct their own due diligence before making any decisions based on this information.
- Except for the sections on XLink, this technical report is based entirely on the keynote presentation delivered by Panmnesia at the 2024 Summer Conference of The Institute of Semiconductor Engineers in August.

panmnesia
주식회사 파네시아

| 공식 홈페이지 |

panmnesia.com

| 이메일 |

contact@panmnesia.com

| 선행개발캠퍼스 |

대전광역시 유성구 한밭대로371번길 42

| 사업개발캠퍼스 |

서울특별시 영동대로85길 38 진성빌딩 10층

링크드인 페이지

유튜브 채널